Computer Aided Grid Interface

An interactive CFD Pre–Processor

Final Report

(performance period: 16 May 1991 – 15 May 1996)
Technical Monitor
Mr. Ted Benjamin

Sponsoring Agency
CFD Branch
NASA Marshall Space
Flight Center
MSFC, AL 35807

Principal Investigator:

Dr. Bharat K. Soni
Professor, Aerospace Engineering

Table of Contents

Summary

NASA maintains an applications oriented computational fluid dynamics (CFD) efforts complementary to and in support of the aerodynamic–propulsion design and test activities. This is especially true at NASA/MSFC where the goal is to advance and optimize present and future liquid–fueled rocket engines. Numerical grid generation plays a significant role in the fluid flow simulations utilizing CFD. An overall goal of the current project was to develop a geometry–grid generation tool that will help engineers, scientists and CFD practitioners to analyze design problems involving complex geometries in a timely fashion. This goal is accomplished by developing the *CAGI*: Computer Aided Grid Interface system. The CAGI system is developed by integrating CAD/CAM (Computer Aided Design / Computer Aided Manufacturing) geometric system output and / or Initial Graphics Exchange Specification (IGES) files (including all the NASA–IGES entities), geometry manipulations and generations associated with grid constructions, and robust grid generation methodologies. This report describes the development process of the CAGI system.

Introduction

A multitude of techniques and computer codes have been developed for generating computational grids in arbitrary regions. However, in most of these codes and methodologies, the evaluation of the geometry input and realization of mapping between physical and computational space allowing appropriate zonal/block strategies are long, very laborious, and extremely time consuming. Geometry–grid generation is considered as the most time and cost critical part in the typical CFD application. The development of CAGI was initiated in 1992 to fulfill simulation of complex geometries encountered in propulsion problems at MSFC in a timely fashion for engineering design. CAGI is developed in a modular fashion. A self explanatory pictorial view of different modules and their linkage is provided in Figure 1. The computer languages FORTRAN 77 and C in a Unix environment are utilized for portability. An X–based interface is provided allowing SGI's GL (Graphics Library) or X–Graphics access by command line argument. The dynamic allocation of memory and linked lists on a well–defined data structure is allowed in the CAGI development. The module, IGES transformer is developed to facilitate all critical geometrical entities. All NASA–IGES geometrical entities have been considered in this development. The Non–Uniform Rational B–Spline (NURBS) curve/surface/volume representation is selected for the geometric description. The NURBS offers a control point based parametric representation which is widely utilized for interactive design applications using CAD/CAM systems. An inverse formulation of NURBS is developed for evaluation of control points associated with the sculptured discretized specification of geometrical entities. The application of these developments in surface grid redistribution, adaptation, remapping and optimization are explored.

The final version of the CAGI system offering an IGES transformer, conversion of the geometric entities into standard NURBS data structure and various geometry manipulation and generation was released to NASA/MSFC personnel in April 1996. The IGES transformer module was incorporated in the grid system GENIE++ and the National Grid System developed at the Mississippi State University (MSU). The CAGI system has been applied to various grid generation problems based on the IGES supplied geometric information.

The detailed development of the CAGI is published as a Ph.D Dissertation entitled, "CAGD Techniques in Grid Generation" in December 1996. This dissertation is included under the section on

CAGI development. A user's guide is also included in the same section. This user's guide can be electronically accessed from the following address: http://www.erc.msstate.edu/thrusts/grid/cagi/ A brief description of the computer code GENIE++ is provided in Appendix A-1.
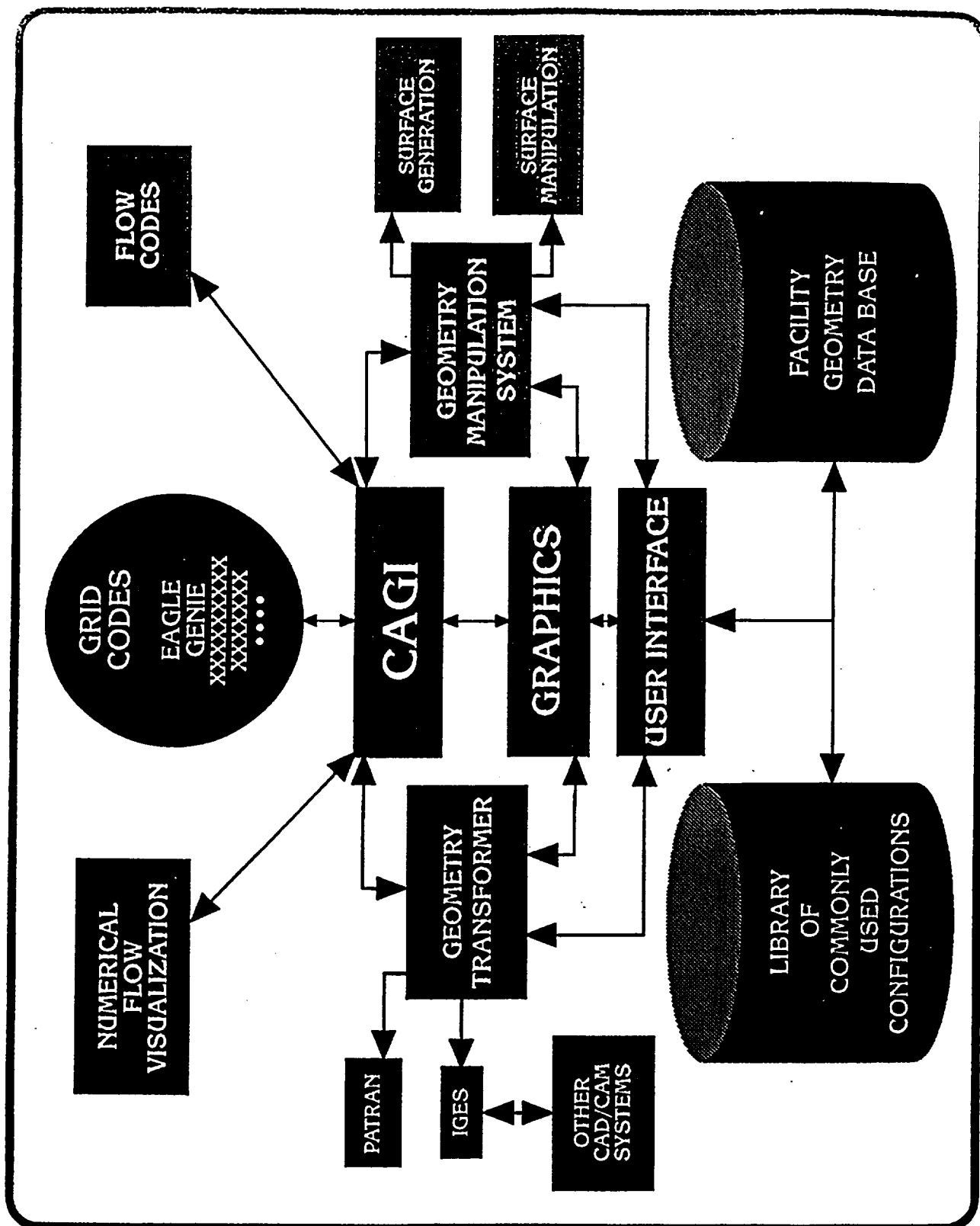
Figure 1.   CAGI Modules

Ph.D Dissertation

CAGD Techniques in Grid Generation

.

CAGD TECHNIQUES IN GRID GENERATION

By

Tzu–Yi Yu

A Dissertation
Submitted to the Faculty of
Mississippi State University
in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy
in Computational Engineering
in the Department of Electrical and Computer Engineering
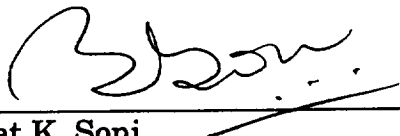
Mississippi State, Mississippi

December 1995

CAGD TECHNIQUES IN GRID GENERATION
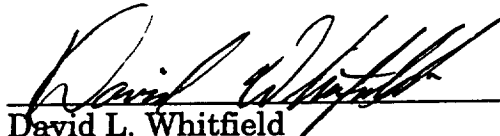
By

Tzu–Yi Yu

Approved:

_____
Bharat K. Soni
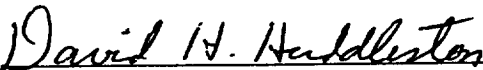Professor of Aerospace
Engineering
(Director of Dissertation)

_____
Joe F. Thompson
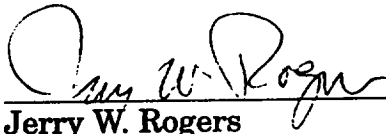Professor of Aerospace Engineering

_____
David L. Whitfield
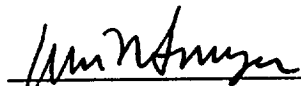Professor of Aerospace Engineering

_____
Jianping Zhu
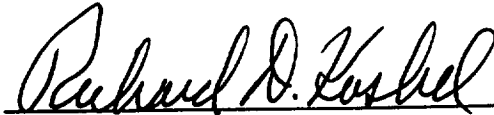Associate Professor of Mathematics

_____
David H. Huddleston
Associate Professor of Civil
Engineering

_____
Jerry W. Rogers
Graduate Coordinator of
Computational Engineering

_____
W. Glenn Steele
Dean of the College of Engineering

_____
Richard D. Koshel
Dean of the Graduate School

Name: Tzu–Yi Yu

Date of Degree: December 15, 1995

Institution: Mississippi State University

Major Field: Computational Engineering

Major Professor: Dr. Bharat K. Soni

Title of Study: CAGD TECHNIQUES IN GRID GENERATION

Pages in Study: 187

Candidate for Degree of Doctor of Philosophy

The objective of this study is to develop the algorithms for static and dynamic grid generation applicable to complex industrial configurations by utilizing Computer Aided Geometry Design (CAGD) techniques. The Non–Uniform Rational BSpline (NURBS) is used as a basis for the geometric definition and in the development of grid generation schemes. The algorithms which bridge the gap between CAD/CAM systems and grid generation systems have been developed by utilizing the IGES (Initial Graphics Exchange Specification) file format, which is the output of the CAD/CAM systems. The IGES and NASA/IGES prescribed geometric entities have been supported by developing algorithms which transfer all geometric entities into a common NURBS data structure. These robust transformation algorithms developed in this study provide enhancements and generalizations to the existing CAGD techniques pertinent to the grid generation process. The algorithms to interactively construct NURBS curves, surfaces and volumes widely applicable to CFD configurations are presented. Projection and inversion techniques applicable to an interpolated sculptured discretized data set are developed and validated. The reparameterization algorithms to overcome the influence of NURBS geo-

metric characteristics on the quality of the resulting grids have been developed. These algorithms have been cast into a computer software *CAGI* (Computer Aided Grid Interface) to facilitate the geometric modeling and surface preparations associated with the grid generation process. The *CAGI* software provides a menu driven interactive environment based on the *FORMS* and *GL* graphics libraries. The interactive graphics capabilities of *CAGI* have been applied to initiate the surface grid generation associated with trimmed surface entity (IGES entity 144). The NURBS based griding algorithms have been applied to construct dynamic grids associated with complex solution adaptive and temporally deforming configurations. Computational examples demonstrating the success of these algorithms in allowing the treatment of CAD / IGES files, formulating very concise NURBS control polygon (control net or control volume) associated with industrial geometries, and addressing grid redistribution, solution adaptation and dynamically deforming grids are exercised. The algorithms developed in this study along with efficient computer memory requirement and fast evaluation of NURBS entities make the *CAGI* program very attractive for addressing grid generation needs associated with complex industrial configurations efficiently and economically.

DEDICATION

To my family

ACKNOWLEDGMENTS

Koomullil and Sujay Shaunak for their reading and corrections of this dissertation.

Finally, the greatest acknowledgement is given to my parents, my sister and brother, for their patience, understanding, support and encouragement; without their support and love, I could not complete this work.

<div align="right">

Tzu–Yi Yu

Starkville, Mississippi

December 1995

</div>

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER I

INTRODUCTION

With the advent of supercomputers and high powered workstations, computational field simulation has become a more common exercise in modern analysis, design and manufacturing. This simulation procedure can be divided into three steps: grid generation (pre–processing – generation of a discrete representation of a surface or volume for the solution domain), solution algorithms (processing – numerical solution of equations of fluid mechanics), and the scientific visualization (the post–precessing – interpretation of simulated physical field characteristics). Numerical grid generation is usually the most labor intensive part of any Computational Field Simulation (CFS) application. In fact, at present it can take significantly more labor time to construct a typical CFS grid than it does to execute the flow simulation code on the grid or to analyze results. A multitude of techniques for grid generation with increasing capability has been developed over the past decade [Ref 57, 59, 63 ~66]. Among the more notable achievements are elliptic equation based smoothing procedures [Ref 71~75] and algebraic interpolation schemes [Ref 57, 60~63] for the generation of surface and volume grids. However, 80%~90% of the grid generation labor time is spent on geometry processing. For a routine application of CFS, in an industrial environment, the overall response time for CFS must be reduced considerably. As noted by Ives [Ref 36]:

"...... The industrial requirement is for reliable one hour grid generation

1

turnaround for one–time geometries when run by designers. The system must include CAD–to–grid links which resolve tolerance issues and produce grids with a quality good enough for the flow solver. The designer has to feel that the grid generation processes is under control and is predictable."

In many of today's industrial applications, most of the geometrical configurations of interest to practical problems are designed using a CAD/CAM system. As pointed out by the NASA Steering Committee on Surface Modeling and Grid Generation [Ref 6,35,89], the linkage between the CAD/CAM systems and the grid generation systems will significantly reduce the overall turn around time for CFS applications. Unfortunately, there are many different geometry output formats which force the designer to spend a great deal of time manipulating geometrical entities in order to achieve a useful sculptured geometrical description for grid generation. In addition, there is a danger of losing the fidelity of the geometry in this process of data transfer between different I/O formats [Ref 6,88]. The other issue related to field simulation is the grid quality. It is well known that the quality of the grid affects the accuracy of the solution and the computation time [Ref 1, 73]. It may be necessary to reconstruct the grids for a more satisfactory result after obtaining the first solution. This reconstruction procedure involves a change of either resolution (the size of the grids) or the spacing (the distance between grid points) functions. However, this process, especially true for a complex grid, is tedious and very time consuming. This provides the basic motivation for this study.

To bridge the gap between the CAD/CAM systems and grid generation systems, it is necessary to establish the communication paths so that the geometries and grids defined within these two systems can be linked with each other. The Initial Graphics Exchange Specification (IGES) [Ref 35] is a widely

accepted standard for the geometry exchange. All CAD/CAM systems support the IGES format as an Input/Output of resulting geometries. Therefore, in this study, IGES format has been considered as a standard format for geometry specification. The description of IGES format is provided in the following section. Regarding the grid quality and efficiency in geometry / grid manipulation, it is essential to develop grid generation schemes in conjunction with well defined analytical / semi-analytical sculptured (parametric) geometric representation. This parametric geometry definition must facilitate the generation / manipulation / refinement of the high quality geometry (grids). The Non-Uniform Rational BSpline (NURBS) [Ref 18,43~47] has been selected as the standard for parametric geometric representation in this study. The attributes and characteristics of the NURBS are provided in the following section. Hence, in this study, the IGES file I/O format and the NURBS representation are utilized as the standard for CAD/CAM system output and geometric definition respectively.

<u>IGES Format</u>

The IGES format was defined by researchers and engineers from industry, government organization and academics in the late 1970's and early 1980's. This format is intended to describe every aspect of a geometric data set (such as the coordinates of the data set, what color to be plotted and even the line width to be plotted ...), and is designed to be the national standard of Input/Output format for CAD/CAM systems. There are five sections in each IGES file, namely, the Start section, Global section, Directory entry section, Parameter data section and the Terminate section. All entities described in an IGES file are labeled with an integer number (for example, a circular arc is labeled with 100) and classified into four classes. These classes are "curve and

surface geometry entities", "constructive solid geometry entities", "annotation entities" and "structure entities". Among these four classes, the widely utilized class for CFS engineers and designers is the "curve and surface geometry entities". These entities are labeled from 100 ~ 199. An example of an IGES file describing the definition of a sphere is listed in Appendix A. In this IGES example, the Start section includes the human–readable prologue to the file. All records in this section shall have the letter "S" in column 73 and a sequence number in columns 74 through 80. The Global section of this file contains the information describing the preprocessor and information needed by the postprocessor to handle the file. All records in this Global section shall contain the letter "G" in column 73 and a sequence number in columns 74 through 80. Many important parameters are defined in this section, two of which are the characters of parameter delimiter and record delimiter. The default characters are the "comma" and "semicolon" respectively. The Directory entry section has one directory entry for each entity in the file. This directory entry for each entity is fixed in size and contains twenty fields of eight characters each, spread across two consecutive eighty character lines. All records in this section contains the letter "D" in column 73 followed by a 7 digit sequence number. For this example, the IGES file describes a sphere defined by a NURBS surface; hence, the entity number 128 is shown in the first and eleventh fields of this directory entry. The Parameter data section contains the parameter data associated with each entity defined in the Directory entry section. For this example, the NURBS information, such as the control vertices, the knot vector and the *weights*, are defined in this section. The Terminate section is the last section. It contains a character "T" in column 73. There is only one line of the Terminate section in an IGES file. This section describes

the number of lines used in the Start section, Global section, Directory section and Parameter data section in the IGES file.

As aforementioned, the IGES file is an attempt to represent every aspect of geometry data, and thus its documentation is long and difficult to comprehend. Utilization of the IGES data file requires expert knowledge of the data format and the interpretation associated with the geometry description of respective entities. Hence, even though this format has been defined since 1980, it has not been widely used by the grid generation community. To enhance the utilization of an IGES file and to expedite the entire grid generation process, scientists at various NASA centers (NASA Ames, NASA Langley and NASA Lewis) have formed an IGES committee. This committee, referred to as the NASA IGES committee, selected several commonly used entities in CFS simulation as a group. This group of IGES entities is known as the NASA–IGES. NASA–IGES provides a common standard for geometric description between the CAD/CAM and the CFS community. All these efforts simply show the importance of the IGES format in creating quick and efficient results for the geometry transformation.

In view of these efforts, IGES has been selected as the CAD/CAM system for geometry Input/Output in this study. The algorithms to interpret, analyze and manipulate the IGES geometric entities by utilizing the NURBS as a standard geometric representation have been developed in this work.

NURBS

NURBS stands for Non–Uniform Rational BSpline and is a parametric form consisting of the control polygons, knot vectors, *orders* and *weights* to represent the geometric entity accurately. If all the *weights* of a NURBS entity are equal to one, then it is called a Non–Rational BSpline (or simply

BSpline). The BSpline was developed in the 1970's and 80's by Carl de Boor, M. G. Cox and J. Schoenberg. In 1973, R. Riesenfeld presented the paper entitled "Applications of BSpline approximation to geometric problems of computer-aided design" which was the first application of the BSpline applied to CAD. The properties and applications of NURBS were initialized and extensively developed during the 1980s. The first commercial product which used the NURBS to represent the geometry came from the SDRC (Structural Dynamics Research Corp.) in 1983. The Boeing company proposed the NURBS as an IGES standard in 1981, and now the NURBS curve and NURBS surface have been adapted as the IGES geometric entities 126 and 128.

Recently, the IGES entities 126 and 128 have become increasingly popular in geometric definition. This popularity is due to various characteristics of the NURBS, especially the shape preserving property. Many commonly used curve and surface definitions can be analytically represented by NURBS. These commonly used geometric entities include the circular arc, conic arc, line, cubic spline curve, cylinder, ruled surface, surface of revolution, parametric surface ... and so on. In addition to this shape preserving property, NURBS has many other powerful features, such as the convex hull, local control, variation diminishing and affine invariance. The convex hull property ensures that the NURBS curve (surface or volume) will lie entirely inside its associated control polygon (control net or control volume) [Ref 23~25,34]. The local control property facilitates the local modification of the geometry without altering the entire geometric shape. This attractive property makes it possible to refine / reconstruct the desired portion of a geometry locally. The variation diminishing property, taking a NURBS curve as an example, is defined such that if a straight line intersects the NURBS control polygon of a planar

NURBS curve k times, then it can intersect the NURBS curve at most k times. This property ensures that the NURBS is always convex whenever its control polygon is convex, and the NURBS curve can be convex even when its control polygon is not convex. Furthermore, if a NURBS curve has a point of inflection, then its associated control polygon must have at least one point of inflection [Ref 23,34]. This property is useful in predicting the shape of a NURBS representation.

Also, the NURBS geometry tool kits which include the knot insertion, degree elevation and splitting algorithms, make NURBS useful in their own right. The knot insertion algorithm is used to increase the associated knot vector without changing the original NURBS shape. After increasing the number of knot values in a knot vector, the designer has more freedom to modify the associated NURBS curve (surface or volume). Similar to the knot insertion algorithm, the degree elevation provides the flexibility to modify the NURBS entity by increasing the associated degree without changing the original geometric shape. These two functions are important, fundamental and unique to the NURBS representation. The splitting algorithm allows the designer to "cut" a NURBS entity into two different NURBS entities.

All these geometric properties make the NURBS stable for representing complicated data sets and also very attractive to the designers and engineers in CFS and CAD/CAM. In fact, because of these attractive properties, many efforts have been proposed utilizing the NURBS as the standard and geometric representation. For example, after presenting the the NASA–IGES format, the NASA IGES committee further proposed the *NINO* (NASA IGES NURBS ONLY) standard [Ref 6,35]. The scientists and engineers from the Naval Surface Warface Center and Boeing company combined their efforts

and designed the software package *"DT_NURBS"* [Ref 21]. This software package utilizes NURBS definitions as a standard data structure for geometric entities and provides evaluations and manipulations of NURBS entities. A general purpose grid generation package *"NGP"* [Ref 29,71] developed at Mississippi State University / NSF Engineering Research Center also utilizes the NURBS as the standard database. Widely used CAD/CAM packages like *"PA-TRAN"*, *"I-DEAS"* and *"ICEM CFD"* all claim their products support the NURBS geometric database. Besides the IGES format, many national and international standards, such as the *STEP* and *PHIGS*, recognize NURBS as a standard for geometric design. Even in applications of computer graphics, NURBS is used to model the complicated objects. In addition, NURBS has even been implemented and supported as hardware functions. For example, the *GL* (Graphic Library) [Ref 39] in *Silicon Graphics* platform supports hardware functions such as *"nurbscurve"*, *"nurbssurface"*, *"nurbsc"* and *"nurbss"*, ... and so on. These functions account for the hardware implementation of NURBS curves and surfaces definitions. Similar NURBS functions such as the *"gluGetNurbsProperty"*, *"gluNurbsCallback"*, *"gluNurbsCurve"* and *"gluNurbsSurface"* are supported by *OpenGL* [Ref 39] – another popular graphics library. Even in virtual reality applications, NURBS has been used for modeling scenes. In fact, the NURBS representation is becoming the "defacto" standard for the geometry description in most modern CAD/CAM, grid generation systems and computer graphics.

The application of NURBS to grid generation was first presented by Yoon [Ref 81] in 1991. However, in his study, most of the concentration was placed on the BSpline (by considering the case of all *weights* to be one). Following Yoon's work, Yang utilized NURBS for the application of grid adapta-

tion [Ref 83] in 1993. Shih applied the NURBS for data interpolation in the turbomachinery [Ref 53] in 1994, Craft used it for re–splining the surface grids for a CFS analysis of a complete aircraft [Ref 11,67], and Boyalakuntla used it for simulation of the temporally deforming geometries [Ref 13]. These researchers have contributed in utilizing the NURBS interpolation and evaluation routines developed under this study in the area of numerical grid generation. However, the NURBS has been used only as an interpolation tool by these researchers. As a matter of fact, those researchers only utilize the BSpline definition instead of NURBS. In this study, the detailed analysis and algorithms to utilize NURBS in grid generation are presented.

As is mentioned, the communications between the CAD/CAM and grid generations are crucial. In order to communicate the geometry from grid generation to CAD/CAM, the grid generation system must have the ability to read in / output to IGES file. The transform of those geometric entities, which are defined in the IGES standard, to discrete grid formats (such as *PLOT3D* format) has been done in [Ref 86, 87]. As noted, these discrete grids formats have several drawbacks. They will lose the parametric definition, and the grid is difficult to re–evaluate or change the spacing. Also, this discrete grid format can not communicate to a CAD/CAM system anymore. However, many of the geometric representations can be transformed to NURBS analytically. And the NURBS can be used in grid generation to model various geometric definitions, it is possible to communicate with the CAD/CAM system through the IGES entities 126 and 128 which are the NURBS curve and NURBS surface. Hence, the NURBS representation provides the linkage between the grid generation and CAD/CAM system.

This study concentrates on the development of generalized algorithms so that the NURBS can be fully utilized in grid generation system, and thus, eliminate the communication hurdle. The objectives and the organization of this study follows.

Objective and Organization

In this study, the objectives are to develop the algorithms to bridge the gap between CAD/CAM systems and numerical grid generation systems. The CAGD techniques have been utilized in CAD/CAM systems for decades. Many of them are well documented in the related literature. Several existing algorithms have been adopted in this research. These basic and fundamental algorithms include the (NURBS) knot insertion, degree elevation, splitting / joining algorithms [Ref 8,42-47, 76,77] and the evaluation of the BSpline basis functions [Ref 18]. Other existing algorithms, such as the data reduction routines [Ref 38] and the *FORM* Library [Ref 39] which are available in the public domain, have also been utilized in this study.

This work is carried out by first including the development of algorithms for transforming Non–NURBS entities encountered in standard IGES format to NURBS definitions and then following the development of algorithms for easy and efficient NURBS manipulation. The transformation algorithms presented in this study provide the enhancements and generalizations of those described in CAGD literature [Ref 7,42,43]. For example, the transformation algorithm associated with a circular arc to a NURBS curve is enhanced to facilitate an arbitrary arc with different sector angle (the difference of ending angle and starting angle) without extra computation of using the knot insertion algorithm [Ref 45]. The transformation of the conic arc to a NURBS curve representation is generalized by solving an implicit equation

which provides more information necessary for grid generation process. This information also includes the types of the arc, the orientations, the sector angle and the semi–major and semi–minor axis. Also, the transformation algorithm developed for the surface of revolution is not limited to the full revolution which is the only case described in CAGD literature.

In addition to the transformation algorithms, the NURBS generation algorithms are also developed. In the CAD/CAM and CAGD area, the NURBS is only used for generating the curves and surfaces. However, the NURBS curve and surface algorithms have been extended to the volume generation in view of the application to grid generation. These new algorithms can be applied for various 3D NURBS volume generation algorithms tailored for many CFS related configurations. Also, the modeling techniques for the superelliptic arc and cascading surface by NURBS curve and surface presented in this study have not been discussed before in the literature. These algorithms can facilitate the use of CAD/CAM data for CFS analysis. The most commonly used NURBS evaluation algorithm discussed in related literature is the "de Boor" method [Ref 23, 34]. In this study, a new evaluation algorithm for NURBS representations (curve, surface and volume) is developed. This new algorithm provides a competitive alternative to the de Boor algorithm in terms of the memory used and the computational time. Since NURBS is a parametric representation, the determination of proper parametric values for the desired (smooth) grids on NURBS entities is a difficult issue. The new reparameterization algorithms developed in this study provide an efficient approach to solve this difficulty.

The grid adaptation is frequently used in many CFS applications. It provides the ability for obtaining more accurate solutions. Since the repara-

meterization algorithms can be used for precise grid distribution control on the NURBS entities, this reparameterization algorithm is then used to enhance the adaptation algorithm developed by Yang [Ref 83 ~85] in this study.

The organization of this dissertation is as follows. First, the background and the motivation of this study are introduced in Chapter One. The NURBS descriptions along with the algorithms for transforming curves and surfaces to NURBS representations are described in Chapter Two. The algorithms for generating 3D surface and volume grids by various approaches utilizing NURBS control net and control volume are presented in Chapter Three. The general BSpline interpolation algorithm along with the projection and inversion methods follow. The NURBS re–parameterization algorithms are described in Chapter Four. These algorithms are designed to maintain the discontinuity of the NURBS entities and accomplish a better grid distribution on the physical NURBS entities. A new algorithm developed for a fast and efficient NURBS evaluation is also included. The applications of NURBS in dynamic grid generation are demonstrated in Chapter Five. These applications include the grid adaptation and temporally deforming geometry. In this chapter, the NURBS "local control" property and the re–parameterization algorithm are fully demonstrated. The grids are modeled with a concise NURBS control net (or control volume), and computational examples are included. The overview of the computer software package *CAGI* is presented in Chapter Six. The modules of *CAGI* and the associated functionalities are introduced. There are many packages which claim the capability of reading the IGES file for numerical grid generation. However, very few of them can handle two complicated and difficult entities — the bounded surface (entity 143) and the trimmed surface (entity 144) for "structured" grid topology. The approaches

and strategies of handling these two entities and the examples are shown in
Chapter Seven. The summary and conclusions are listed in the last chapter.

CHAPTER II

NURBS AND THE TRANSFORMING ALGORITHMS

NURBS Formulation

The definitions of NURBS curve, surface and volume are presented as follows:

A NURBS Curve of order k is defined as:

$$C(t) = \frac{\sum\limits_{i=0}^{n} W_i\, d_i\, N_i^k(t)}{\sum\limits_{i=0}^{n} W_i\, N_i^k(t)} \qquad (2.1)$$

where the d_i $i=0,...,n$ denotes the deBoor control polygon and the W_i are the weights associated with each control point. The $N_i^k(t)$ is the normalized BSpline basis function of order k and is defined over a knot vector $T = T_i$ $i= 0,$ ... , $n+k$ by the recurrence relations as shown in equation (2.2).

$$N_i^k(t) = \frac{(t - T_i)N_i^{k-1}(t)}{T_{i+k-1} - T_i} + \frac{(T_{i+k} - t)N_{i+1}^{k-1}(t)}{T_{i+k} - T_{i+1}}$$

$$N_i^1(t) \; \begin{cases} = 1 & \text{if } \; T_i \le t < T_{i+1} \\ = 0 & \text{otherwise} \end{cases} \qquad (2.2)$$

Throughout this study, it is assumed that the knot vector has the form $T = \{0,$ ... , $0, T_k, ... , T_n, 1, ... , 1\}$ with the multiplicity k for the knot value 0 and 1 on both ends of the knot vector. If the knot vectors do not match this format, the knot insertion [Ref 8] technique must be used to achieve the multiplicity of k

14

on the ends of the knot vector, and if the end knot values are not 0 and 1, the knot vector must be normalized by the last knot value to match this format. Because shifting and scaling (normalizing) the knot value will not alter the shape of geometry, the basis function list in equation (2.2) is defined as "Normalization" basis function.

The NURBS surface is the extension of the curve from 1D to the 2D tensor product parametric surface and is shown as equation (2.3).

$$S(s,t) = \frac{\displaystyle\sum_{i=0}^{m}\sum_{j=0}^{n} W_{ij}\, d_{ij}\, N_i^{k1}(s)\, N_j^{k2}(t)}{\displaystyle\sum_{i=0}^{m}\sum_{j=0}^{n} W_{ij}\, N_i^{k1}(s)\, N_j^{k2}(t)} \tag{2.3}$$

Where d_{ij} denotes the 3D control net and W_{ij} are the *weights* associated with each control point. The $N_i^{k1}(s)$, $N_j^{k2}(t)$ denote the normalized BSpline basis functions of *order k1* and *k2* over the two knot vector $T_1=T_i$ $i=0,..,m+k1$ and $T_2=T_j$ $j=0,..,n+k2$ in the I and J directions, respectively. The definition of the BSpline basis functions of NURBS surface is exactly the same as for the curve shown in equation (2.2).

The formula for 3D NURBS volume is defined analogous to NURBS surface and is a 3D tensor product form written as:

$$V(s,t,u) = \frac{\displaystyle\sum_{i=0}^{m}\sum_{j=0}^{n}\sum_{l=0}^{p} W_{ijl}\, d_{ijl}\, N_i^{k1}(s)\, N_j^{k2}(t)\, N_l^{k3}(u)}{\displaystyle\sum_{i=0}^{m}\sum_{j=0}^{n}\sum_{l=0}^{p} W_{ijl}\, N_i^{k1}(s)\, N_j^{k2}(t)\, N_l^{k3}(u)} \tag{2.4}$$

The $d_{i\,j\,l}$ form the 3D control volume, and the $W_{i\,j\,k}$ are *weights* associated with each control point. The $N_i^{k1}(s)$, $N_j^{k2}(t)$ and $N_l^{k3}(u)$ are the

normalized BSpline basis functions of *order k1*, *k2* and *k3* over the two knot vectors $T_1 = T_i$ $i=0,...,m+k1$, $T_2 = T_j$ $j=0,...,n+k2$ and $T_3 = T_l$ $l=0,...,p+k3$ in the I, J and L directions (i.e., the s,t,u directions), respectively.

Transforming Procedures

Transforming the Non–NURBS geometric curves and surfaces to NURBS definition is the main topic of this section. To model a NURBS entity, according to the equations (2.1) – (2.4), one should define the control polygons (or control net / volume), *weights*, knot vector(s) and the *order(s)*.

CAD/CAM systems and grid generation systems often utilize different formats which makes transfer of information between systems difficult. The procedure presented in this section has been developed to facilitate this communication. A geometric entity defined in a CAD system can be represented in many ways (for example, cubic parametric spline surface, ruled surface, surface of revolution or extruded surface ...etc.), while still conforming to the IGES standard. It is well known that most geometric definition can be analytically transformed to a NURBS representation. The transformations of most widely used entities to NURBS are described by Piegl [Ref 43,44,46]. However, there are many practical issues which are not covered in the transforming procedures published in those literature. For example, the IGES representation of the implicit conic arc, an important entity, is not contained in those references. The transforming algorithm for a general circular arc (a circular arc with arbitrary starting and ending points) is also missing from those reference. Another problem is the available literature may not provide sufficient detail. For example, procedures for transforming a surface of revolution into a NURBS are provided only for a 360^0 revolution, but many grid generation applications require a specified range, such as 60^0. The procedures for trans-

forming a ruled surface to a NURBS are only covered for sequence connection, but the IGES format defines two ways of connections [Ref 35]. Also, several transforming algorithms are never discussed in any of the literature. For example, the Transfinite Interpolation (TFI) for a NURBS volume and the modeling of the super–ellipse as the NURBS curve. Hence, the following sections provide enhancements and generalizations to existing transformations developed to meet needs arising from the grid generation process for complex geometries defined in a CAD/CAM system.

Transform the Curve Entities to the NURBS Representations

The algorithms of transforming various curve definitions to NURBS curve representation follows. There are six different cases discussed in this section.

Transform Straight Line (Entity 110) to NURBS Curve

The first (and the easiest) entity to transform is the straight line (entity 110). In the IGES documentation, a straight line is defined as the connection between two 3D data points. Therefore, with regard to NURBS, the definition of the line uses the original two points as the two control points (d). Then, set *order* (k) equal to 2, set *weights* (W) all equal to 1, and set the knot vector as (0,0,1,1). Since there are only two control points, the n is equal to 1. One may argue the need for using NURBS to represent a straight line, because the NURBS form needs extra storage than the traditional straight line definition. This is why even the NASA–IGES committee feels the straight line definition could remain unchanged. However, there are two reasons to transform this entity to NURBS form. First, transforming all entities to NURBS form keeps a simplicity of the database. Second, if this straight line is one of the constitu-

ent entities of the composite curve (refer the "Transform composite curve to NURBS curve" for detail), then representing that composite curve with NURBS form requires that the straight line be represented in NURBS definition so that one can perform knot insertion or degree elevation to join all constituent curve entities into one composite NURBS curve.

Transform Circular Arc (Entity 100) to NURBS Curve

A circular arc (entity 100) as defined in the IGES standard is represented by a center point, starting point and ending point within a given constant Z plane. The two end points and the center point form an arbitrary sector angle which does not necessarily start from zero. It has been shown that any circular arc with sector angle less than or equal to 90° can be represented by NURBS [Ref 42,45]. The basic control polygon for this NURBS representation is shown in Figure 2.1.
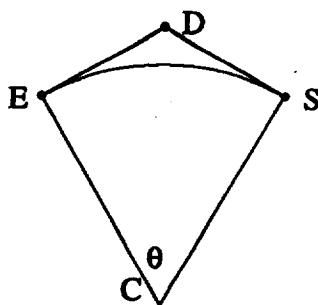
Figure 2.1 The basic control triangle for a circular arc.

In Figure 2.1, C is the center point, S is the starting point and E is the ending point. The sector angle SCE (θ) is less than or equal to 90°. The two tangent lines SD and ED intersect at D. The order of this control polygon is three, with the control points S, D, E (hence, the n is 2) and the weights are 1, $\cos(\theta/2)$ and 1 respectively. The associated knot vector is (0,0,0,1,1,1). A circular arc with sector angle greater than 90° and less than or equal to 180° can be

represented by two arcs with one half of the original sector angle. For each of these two sections the previous procedure can be used to evaluate the corresponding control polygon. A 180° circular arc represented by two control polygons is illustrated in Figure 2.2.
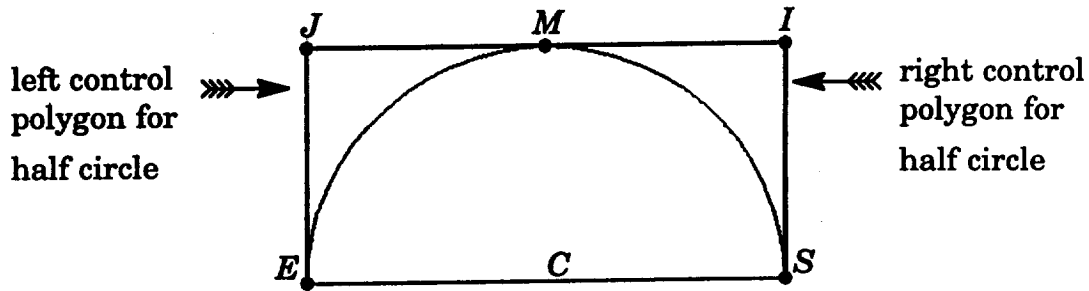
Figure 2.2 The NURBS control polygon for a semi–circle.

These two control polygons can be combined and the common point M can be eliminated. The resulting NURBS information is setting the control polygon to $SIMJE$ (hence, the n is 4), the knot vector to (0, 0, 0, 1/2, 1/2, 1, 1, 1) and the *weights* to (1., $\cos(\theta/n)$, 1., $\cos(\theta/n)$, 1.). A similar procedure can be used for circular arcs between 180° and 270° (with n equal to 6) resulting in a final knot vector of (0, 0, 0, 1/3, 1/3, 2/3, 2/3, 1,1,1) and *weights* (1., $\cos(\theta/n)$, 1., $\cos(\theta/n)$, 1.$\cos(\theta/n)$, 1.), and a knot vector of (0, 0, 0, .25, .25, .5, .5, .75 , .75 ,1,1,1) and weights (1., $\cos(\theta/n)$, 1., $\cos(\theta/n)$, 1., $\cos(\theta/n)$, 1.$\cos(\theta/n)$, 1.) for arcs between 270° and 360° for n equal to 8. These four cases are shown in Figure 2.3.

This approach handles all possible circular arcs with no extra computation (such as knot insertion) involved. Furthermore, the parameterization (distribution on the curves, see detail in Chapter four) is good for all cases.
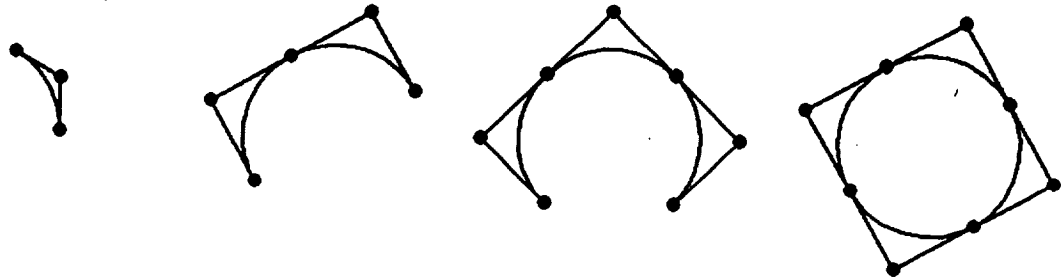
Figure 2.3 Arbitrary circular arcs with the NURBS control polygons.

Transform Conic Arc (Entity 104) to NURBS Curve

The transforming procedure for conic arc was discussed in [Ref 42,44], where they described the case of 3 given control points and changing the weight (conic shape factor) to produce a different family of conic arcs (elliptic, hyperbolic or parabolic arc). However, that case is complete different than the one defined in IGES format. The conic arc defined in IGES is represented by an implicit form $Ax^2 + Bxy + Cy^2 + Dx + Ey + F = 0$, with starting point S and ending point T supplied (counterclockwise). The transforming procedure for a basic conic arc is illustrated in Figure 2.4. In this figure m is the middle point of line TS.
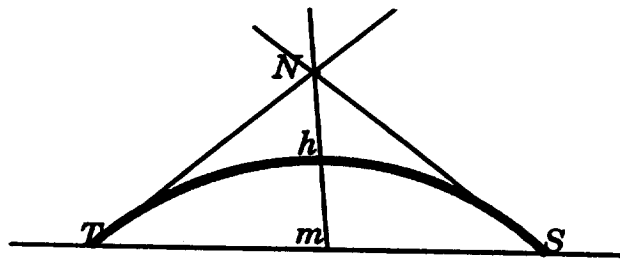
Figure 2.4 Basic NURBS control polygon for a conic arc.

Since the two end points are known, the two slopes of the tangent lines at the end points can be obtained. The equations describing these two tangent

lines can be formed and the intersection point N can be determined. This is accomplished as follows:

Differentiate the implicit form of the conic equation to obtain $2Ax + By + Bxy' + 2Cyy' + D + Ey'=0$. Solving this equation for the derivative yields:

$$y' = (2Ax + By + D) / (-2Cy - Bx - E) \tag{2.5}$$

Substitution of the coordinates of the two end points S and T into equation (2.5) yields the two desired straight lines. The shoulder point h can then be obtained by solving for the intersection of the line Nm and the given implicit equation. The control triangle is then defined by the polygon SNT (hence, the n is 2 for this case) with *weights* of $(1, (mh)/(hN), 1)$. The *order* can be set to 3 and knot vector is defined in a manner analogous to the circular arc. As long as this basic control triangle can be found, the procedure used for the circular arc with the sector angle greater than 90° can be applied to conic arc by simply combining the different control triangles together to form the final control polygon and by setting the proper knot vector. The definition of sector angle θ for the conic arc is only applied to the elliptic arc, for the parabolic or hyperbolic arcs, three control points are sufficient to form the control polygon. Hence, for parabolic or hyperbolic arc, the knot vector is always (0., 0., 0., 1.0, 1.0, 1.0) with n equal to 2. Figure 2.5 shows different conic arcs represented by the NURBS using this algorithm. From left to right, (I): Elliptic arc with equation $2x^2+4xy+5y^2-4x-22y+7=0$, form by two NURBS control polygons. (II): Parabolic arc with equation $4x^2 - 4xy + y^2 - 2x - 14y + 7 = 0$, (III). Hyperbolic arc with equation $2x^2 + 4/3xy - 2y^2-16=0$.
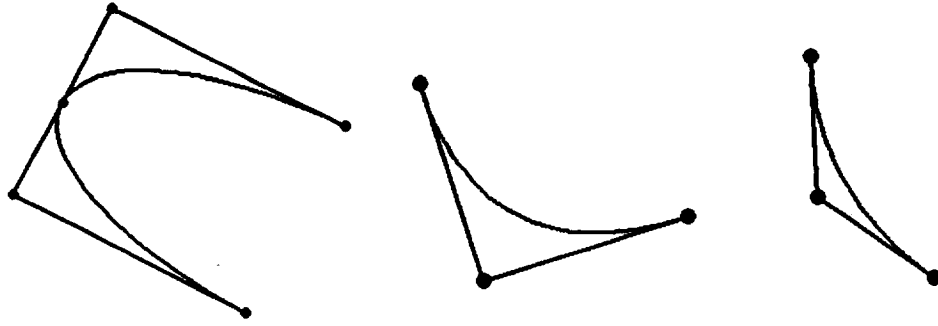
.Figure 2.5 NURBS control polygons represent different conic arc.

Transform Parametric Curve (Entity 112) to NURBS Curve

The cubic parametric curve defined in IGES format is a sequence of parametric polynomial segments. More precisely, it is composed of N $(N \geqslant 1)$ pieces of cubic parametric segments as illustrated in Figure 2.6.
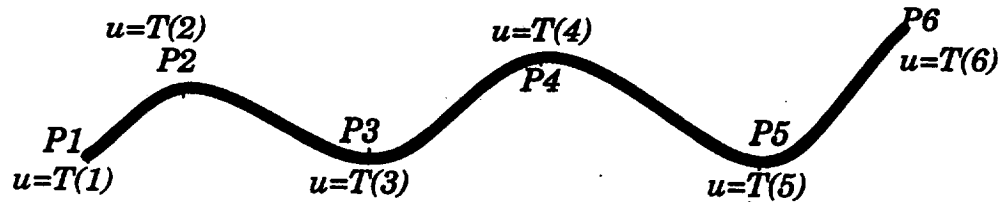
Figure 2.6 The definition of parametric curve in IGES format.

In Figure 2.6, the $T(i)$, $i=1, .. , N+1$ are the breakpoints. For this case, $N=5$, hence, there are 5 cube parametric segments constitute the final curve. For each parametric curve, it is defined as

$$C(u) = a + b\,t + c\,t^2 + d\,t^3 \quad T(i) \leqslant u \leqslant T(i+1) \text{ and } t = u - T(i) \qquad (2.6)$$

It has been proven [Ref 23,24,34] that the cubic Bezier curve is a special case of a BSpline curve with knots vector of (0, 0, 0, 0, 1, 1, 1, 1) (no interior knot value). Also, the BSpline curve is a special case of NURBS curve with all weights equal to 1. The mathematical transformation from parametric cubic spline curve in IGES definition to NURBS is accomplished as follows:

The matrix form of the each simple cubic parametric curve, according to equation (2.6), can be expressed as $C(t) = [1\ t\ t^2\ t^3]\ I_{4x4}\ [a\ b\ c\ d]^T$ where I_{4x4} is the identity matrix and $[a\ b\ c\ d]^T$ is the transposed matrix containing the coefficients of the cubic curve. The matrix form of the cubic Bezier curve is expressed as $C(t) = [1\ t\ t^2\ t^3]\ B_{4x4}\ [b_0\ b_1\ b_2\ b_3]^T$. The B_{4x4} is the cubic Bezier matrix and $[b_0\ b_1\ b_2\ b_3]^T$ is the transpose matrix containing the Bezier control polygon. The strategy is to first transform the cubic parametric curve to Bezier form, since a Bezier curve can be treated as the special case of a NURBS curve. Each segment of parametric spline curve is transformed to a Bezier curve by finding the associated Bezier control polygon. This is done by setting the the two matrix equations to be equal

$$
\text{Bezier} = [1\ t\ t^2\ t^3]
\begin{bmatrix}
1 & 0 & 0 & 0 \\
-3 & 3 & 0 & 0 \\
3 & -6 & 3 & 0 \\
-1 & 3 & -3 & 1
\end{bmatrix}
\begin{bmatrix}
b_0 \\ b_1 \\ b_2 \\ b_3
\end{bmatrix}
$$

$$
= \text{Cubic curve} = [1\ t\ t^2\ t^3]
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
a_i \\ b_i \\ c_i \\ d_i
\end{bmatrix}
\tag{2.7}
$$

Solving the equation (2.7) for the Bezier control polygon. Since the cubic parametric spline defined in IGES is composed of N pieces of cubic curves, the range of parametric value t for each piece is not the same as for the Bezier curve. Hence, a re–parameterization of the cubic parametric curve is necessary. For each piece of cubic curve, the coefficients $[a_i\ b_i\ c_i\ d_i]^T$ can be obtained from the IGES file, therefore, the final equation to solve (for each segment) is

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \frac{1}{3} \begin{bmatrix} 3 & 0 & 0 & 0 \\ 3 & 1 & 0 & 0 \\ 3 & 2 & 1 & 0 \\ 3 & 3 & 3 & 3 \end{bmatrix} \begin{bmatrix} a_i \\ b_i h \\ c_i h^2 \\ d_i h^3 \end{bmatrix} \tag{2.8}$$

where $h = T(i+1) - T(i)$ and $T(i)$ is the break value defined in the IGES file. After all the Bezier control polygons have been obtained, one can join them together and set the multiplicity of joint knot value equal to 3 to form the final Bspline curve. For example, if two cubic Bezier control polygon are obtained, the final knot vector will be set as (0, 0, 0, 0, 0.5, 0.5, 0.5, 1, 1, 1, 1) and the final curve would be C^0 continuous with *order* equal to 4 and all *weights* equal to 1. The knot removal algorithm described in [Ref 5,76] can be applied to eliminate the redundant knot and reduce the number of control points. Figure 2.7 (not applying the knot removal algorithm) demonstrates this approach.

Figure 2.7 BSpline control polygon for parametric curve with 2 segments.

Transform Composite Curve (Entity 102) to NURBS Curve

A composite curve(entity 102) is defined as a curve entity consisting of lists of constituent curves. The constituent curve can be any parameterization curve except another composite curve. And this entity is a directed curve, which means the direction of the composite curve is induced by the direction of the constituent curves in the following manner: The start point for the com-

posite curve is the start point of the first curve entity appearing in the defining list, and the terminate point for the composite curve is the terminate point of the last constituent curve appearing in the defining list. Within the defining list itself, the terminate point of each constituent curve entity has the same coordinates as the start point of the succeeding curve entity. It is quite difficult to represent the composite curve precisely without transforming all the constituent curves to the NURBS form. After transforming all curve entities (like straight lines, circular arcs, conic arcs, parametric curves and rational BSpline curves), the "NURBS Joining" algorithm for all the constituent NURBS curves is performed to form the NURBS representation for the composite curve. The procedure is illustrated as follows:

Suppose two constituent curves C_1 and C_2 (already transformed to NURBS definition) form a composite curve. Then the first step is to perform the degree of elevation [Ref 14] of the lower degree curve so that the curves can have the same *order*. The second step is to adjust the knot vector of the second curve C_2 so that the first knot value of the second curve can have the same value as the last knot value of the first curve. Shifting the knot vector will not change the original NURBS curve because the basis function is a "normalized" basis function. The third step is to build up the final knot vector by joining the two knot vectors into one knot vector and set that knot value at the joint point to have the multiplicity equal to (*order* − 1). For example, if the first knot vector is [0,0,0,1,1,1] and the second knot vector is [2,2,2,3,3,3], adjust the second knot vector by shifting −1 to each value. Thus, the second knot vector becomes [1,1,1,2,2,2]. Suppose the *order* of these two curves are 3, then, the final knot vector should be [0,0,0,1,1,2,2,2] (one may notice the interior knot 1 has multiplicity of (*order* −1) = 2). The fourth step is to match the

weights by timing the ratio of (the last *weight* of the first curve/ the first *weight* of the second curve) to all the *weights* of the second curve so that the *weight* at the joint point for the two curve are the same. The last step is to build up the final control polygon and *weights* by throwing the first control point and *weight* of the second curve away and jointing the others as one control polygon and one *weights* vector. After these procedures have been applied to all the constituent curves, a composite NURBS curve should be formed. One more procedure that may apply to this final curve is to perform the knot removal to remove the redundant knot vector [Ref 5,76]. Figure 2.8 shows this algorithm for transforming the composite curve consisting of, from right to left, one straight line, one circular arc, one straight line, one ellipse arc and a straight line.
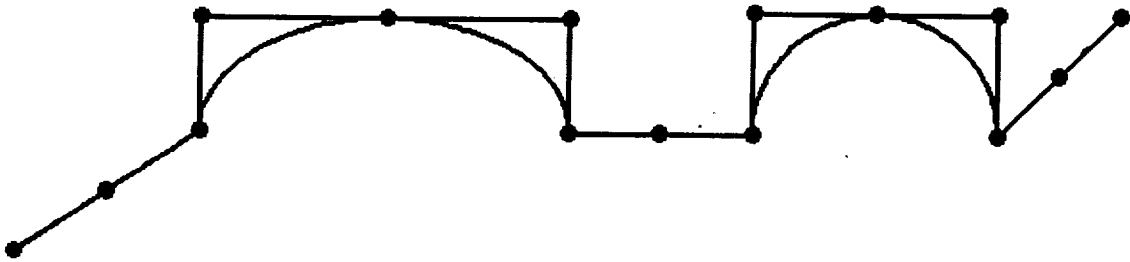
Figure 2.8 NURBS control polygon for a composite curve.

Transform Superellipse to NURBS Curve

A superelliptic arc can be described as the equation (2.9)

$$(\frac{x}{a})^\eta + (\frac{y}{b})^\eta = 1 \tag{2.9}$$

where a is the semi–major and b is the semi–minor axes of the superellipse. Special cases of the equation (2.9) include a circle (with $a = b$, and $\eta = 2$), an ellipse (with $a \neq b$, and $\eta = 2$) and a rectangle (with $a \neq b$, and $\eta = \infty$).

The definition of superellipse is not included in IGES format, however, it is a commonly used geometric description. An example of this is the modeling of a transition duct used for the test of a single–engine nozzle [Ref 41,48]. The transition duct was designed by using a sequence of constant–area, super-elliptic cross sections according to equation (2.9). In most of the literature, the process of obtaining the exponent of the superellipse η was described as solving the implicit function relating the quantities a, b and η to A_{cs} of equation (2.10).

$$A_{cs} = \frac{\Gamma(1/\eta)^2}{\Gamma(2/\eta)}(2/\eta)(4ab) \qquad (2.10)$$

In this study, the transforming of this superellipse to NURBS curve is presented in a more straightforward way as follows.

This transforming approach is a combination of the circular arc and the conic arc algorithms.
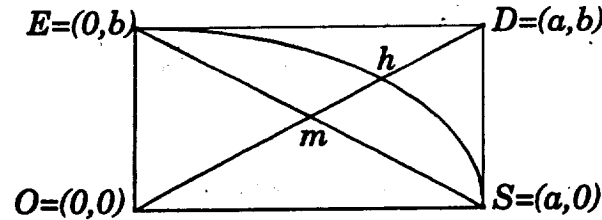
Figure 2.9 Illustration of the NURBS control points for a superellipse.

Consider a superellipse with semi–major a and semi–minor b in the first quadrant shown in Figure 2.9. This arc starts at the point $(a,0)$ and ends at the point $(0,b)$. Two tangent lines intersect at the point (a,b). Similar to the algorithm of circular arc, these three points can be used as the NURBS control polygon while setting the order to be 3 with knot vector (0., 0., 0., 1., 1., 1.). The weights at the starting and ending control polygon can be set to 1.0. The

only problem left is determining the *weight* at the middle point D of the control polygon. This is done similar to the algorithm of the conic arc. Construct the straight line OD and let this line intersect with the line SE and the superelliptic arc at the points of m and h. The *weight* at point D is then set as the ratio of *(hm / hD)*.

This approach is self–explanatory. When the exponent of the superellipse η increases, the arc is changing from a circular arc to a rectangular arc, this means that point h is approaching to the control point D. Also, the distance of hD is decreasing and as a result the *weight* at point D is increased. This situation matches the NURBS theory — a NURBS curve is pulled towards to the control point when the *weight* of this control point increased. The mathematic verification can also be done by comparing the h point with the shoulder point evaluation from NURBS representation. Since the variables *(a, b* and η) of superellipse are all given, the h can be solved from the intersection of the line OD and the arc. On the other hand, after the entire NURBS representation is set up for this superellipse, the shoulder point h can also be evaluated with the parametric value $t = 0.5$. Comparing to the locations of these h's, one can find out that the relative deviation is as small as 1.0e–9. Table 2.1 shows the selected values of the exponent η of the superellipse and the corresponding *weights* values.

Table 2.1 The relationship between exponent η and *weights*.

η	*weight*
2.000000	0.7071067807
2.076143	0.7615055209
2.184741	0.8391550277
2.310944	0.9294727665
2.446475	1.0265482055
2.588168	1.1281144695
2.736506	1.2345144266
2.894152	1.3476587943
3.064489	1.4699782629
3.250206	1.6034070829
3.453315	1.7493976138
3.676614	1.9099667660
3.924127	2.0880154404
4.201364	2.2875017047
4.515468	2.5136151423
4.875638	2.7729511992
5.293192	3.0736854139
5.786112	3.4287875496
6.375087	3.8531827169
7.047038	4.3374610450
7.759080	4.8507150955
8.451551	5.3499221183
9.061041	5.7893464878
9.533431	6.1299460466
9.836925	6.3487773166
9.975085	6.4483977760
9.999865	6.4662654998
10.00000	6.4663630857

From Table 2.1, one can also notice that in the case of circular arc or the elliptic arc (when $\eta = 2$), the corresponding *weight* (for the sector angle equal to 90^0) is the same as $\cos(90^0/2.0)$ which has been discussed in circular/elliptic arc section.

Transform the surface entities to the NURBS representation

The algorithms of transforming various surface definitions to NURBS surface representation follows. There are four different cases discussed in this section.

Transform Cubic Parametric Spline Surface (Entity 114) to NURBS Surface

The cubic parametric spline surface defined by IGES is composed of M by N cubic patches as illustrated in Figure 2.10.
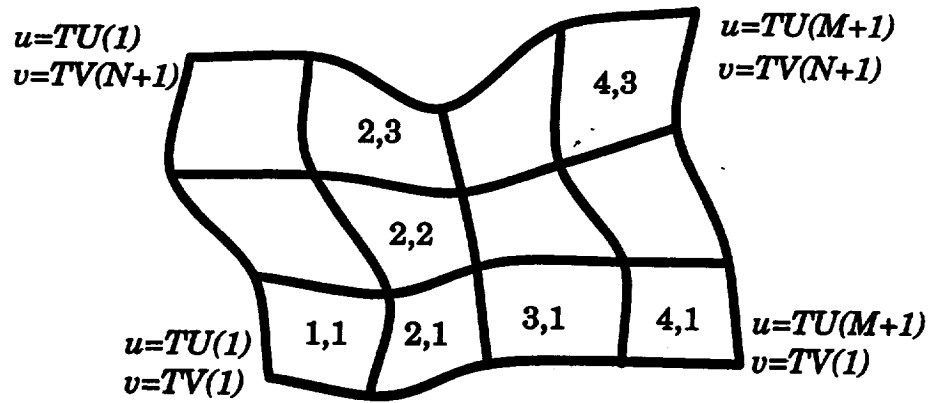
Figure 2.10 Illustration of cubic parametric surface defined in IGES.

The definition of this surface is expressed as:

$$S(u,v) = a + bs + cs^2 + ds^3 + t\,(e + fs + gs^2 + hs^3) +$$
$$t^2\,(k + ls + ms^2 + ns^3) + t^3(o + ps + qs^2 + rs^3) \qquad (2.11)$$

Two breakpoint vectors are $TU(i),...,TU(M+1)$ and $TV(i),...,TV(N+1)$ where $TU(i) \leq u \leq TU(i+1)$ $i= 1, ... , M$ and $s = u - TU(i)$ and $TV(i) \leq u \leq TV(i+1)$ $i= 1,.., N$ and $t = v - TV(i)$.

The strategy for transforming this entity to a BSpline tensor product surface is similar to the one for the cubic parametric spline. The matrix form for the parametric cubic spline surface, according to equation (2.11), can be expressed in a matrix form as shown in equation (2.12).

$$S(u,v) = \begin{bmatrix} 1 & s & s^2 & s^3 \end{bmatrix} \begin{bmatrix} a & e & k & o \\ b & f & l & p \\ c & g & m & q \\ d & h & n & r \end{bmatrix} \begin{bmatrix} 1 \\ t \\ t^2 \\ t^3 \end{bmatrix} \qquad (2.12)$$

While the matrix form of the Bezier surface with Bezier control points $B_{i,j}$ can be expressed as equation (2.13).

$$S(u,v) = \begin{bmatrix} 1 & u & u^2 & u^3 \end{bmatrix} A_{i,j} \ B_{i,j} \ C_{i,j} \begin{bmatrix} 1 \\ v \\ v^2 \\ v^3 \end{bmatrix} \qquad (2.13)$$

$$\text{where} \qquad A_{i,j} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -3 & 3 & 0 & 0 \\ 3 & -6 & 3 & 0 \\ -1 & 3 & -3 & 1 \end{bmatrix} \quad C_{i,j} = \begin{bmatrix} 1 & -3 & 3 & -1 \\ 0 & 3 & -6 & 3 \\ 0 & 0 & 3 & -3 \\ 1 & 0 & 0 & 1 \end{bmatrix}$$

The coefficients of this cubic parametric surface are given from an IGES file, therefore, the variables of equation (2.12) are all known, and the only unknown for equation (2.13) is matrix term of Bezier control points B_{ij}. Hence, the Bezier control points for each bi-cubic patch are obtained by setting equation (2.12) be equal to equation (2.13) and solving the matrix equation (2.14) with necessary re-parameterization.

$$
[B_{ij}] = P \begin{bmatrix} a & eh_2 & kh_2^2 & oh_2^3 \\ bh_1 & fh_1h_2 & lh_1h_2^2 & ph_1h_2^3 \\ ch_1^2 & gh_1^2h_2 & mh_1^2h_2^2 & qh_1^2h_2^3 \\ dh_1^3 & hh_1^3h_2 & nh_1^3h_2^2 & rh_1^3h_2^3 \end{bmatrix} Q
$$

(2.14)

where $\quad P = \dfrac{1}{3}\begin{bmatrix} 3 & 0 & 0 & 0 \\ 3 & 1 & 0 & 0 \\ 3 & 2 & 1 & 0 \\ 3 & 3 & 3 & 3 \end{bmatrix} \quad Q = \dfrac{1}{9}\begin{bmatrix} 9 & 9 & 9 & 9 \\ 0 & 3 & 6 & 9 \\ 0 & 0 & 3 & 9 \\ 0 & 0 & 0 & 9 \end{bmatrix}$

$h_1 = TU(i+1) - TU(i)$

$h_2 = TV(j+1) - TV(j)$

After all Bezier control patches $B_{i,j}$ are obtained, one can join each sub patch to form the final Bspline surface by setting the multiplicity of the knot value at the joint place to 3 in both directions (I and J). The advantage for using this algorithm is that there is no approximation or interpolation work involved. Therefore, the final Bspline surface represents the same geometry as the original parametric cubic spline surface defined in IGES the file. An IGES file generated by a CAD/CAM package is manipulated to demonstrate this algorithm. This IGES file, representing the nacelle of an engine, contains one cubic parametric surface with 34 by 24 sub patches. The final transformed BSpline surface ends up with 103 by 73 control points as shown in Figure 2.11. Similar to the parametric cubic spline curve, this resulting cubic parametric surface is only C^0 continuous (the composition algorithm is simply the inverse of the splitting algorithm, and in the splitting algorithm, knot insertion is repeated until the multiplicity of the knot value is equal to the degree of the curve. The continuity is then defined as $C^{degree - multiplicity}$). According to this algorithm, the $M \times N$ bi-cubic patches will result in ($3M+1$) by ($3N+1$) control points of a single BSpline surface. One can perform the knot removal algorithm to reduce the redundant knots [Ref 5,38,76].
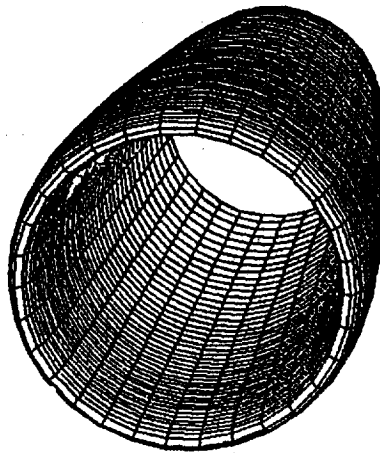
Figure 2.11 BSpline surface represents a nacelle of an engine.

Transform Ruled Surface (Entity 118) to NURBS Surface

As aforementioned, using NURBS to represent a ruled surface has been studied in many papers [Ref 42,43,44]. However, the definition for a ruled surface in IGES format is more general. The IGES defines the ruled surface (entity 118) as one surface formed by moving a line connecting points of either equal relative arc length or equal relative parametric value (the issue of relative arc length and relative parametric value is discussed in Chapter 4) on two curves and by defining a variable named "*DIRFLG*" as direction flag to determine how the surface will be connected. If *DIRFLG* is 0, then the surface will connect the points on the same direction of the two curves. If the *DIRFLG* is 1, the direction of one of the curves will be reversed. The algorithm is described as follows:

Suppose the two boundary curves have been converted to NURBS form according to the previous algorithms. Then, the first step is to make the two knot vectors have the same range by shifting and normalization of all the knot values without changing the multiplicity of any knot value. Next, use the degree raising algorithm to raise the low degree of the curve, which will yield a

new knot vector and new control polygon. If this new knot vector differs from the other knot vector, then perform knot insertion[Ref 8] by inserting the knot value which is not contained in both knot vectors to merge them into one final knot vector. For the J direction, set the *order* equal to 2 and the knot vector as [0,0,1,1]. If $DIRFLG = 0$, then the weights and the control net are set as $W(i,j)$, $d(i,j)$ = (*weights* and control points on the two boundary curve, $i=0..m$ and $j=0,1$). If $DIRFLG = 1$, then the *weights* and the control net are set as $W(i,0)$, $d(i,0)$ = (*weights* and control points of the first curve, $i=0..m$), and $W(i,1)$, $d(i,1)$ = (*weights* and control points of the second curve, $i=m$, $m-1$, ... , 1, 0). Figure 2.12 shows the NURBS rule surfaces with the same boundaries but with different direction flag ($DIRFLG=0$ and $DIRFLG=1$, respectively).

DIRFLG=0   *DIRFLG=1*
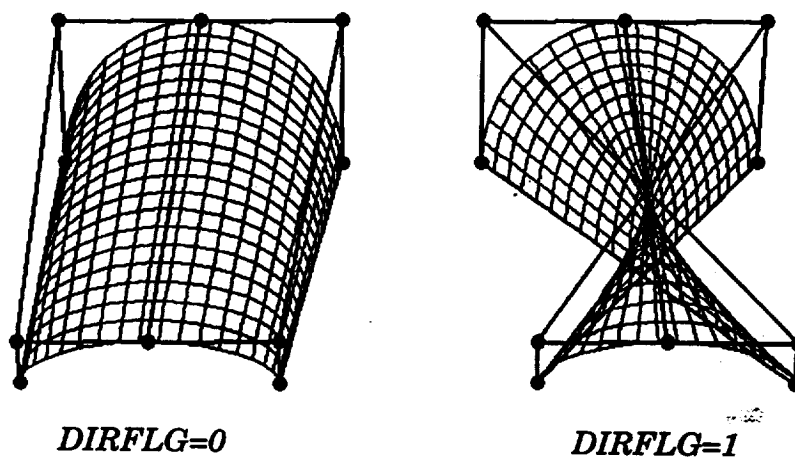
Figure 2.12 NURBS surfaces represent the ruled surfaces.

Transform Surface of Revolution (Entity 120) to NURBS Surface

Again, the surface of revolution has been discussed in many places [Ref 42,43,44,47]. However, the transforming procedures which can cover the more general case is presented. IGES defines the surface of revolution (entity 120) as the surface which is formed by rotating a boundary curve (called genera-

trix) with respect to a straight line (axis of revolution) from a starting angle (not necessarily zero) to an ending angle. This general algorithm can be stated as follows:

First step is translating/rotating the axis of revolution so that it is coincident with the Z axis. It is assumed that the generatrix is defined as NURBS curve with the control polygon d_0 to d_m, *order=k*, *weights* w_0 to w_m. Next, for each control point d_i (on the generatrix $i = 0,..,m$), construct the surface control net d_{ij} j=0 .. n at the i–th cross section according to the starting and ending angle by utilizing the circular arc algorithm. Based on the procedure described in the section of "Transform Circular Arc (entity 100) to NURBS curve", n is determined by sector angle (equal to the difference between ending and starting angle). For example, if the angle is less than 90^0, n is equal to 2, if the angle is in the range of 90^0 ~180^0, n is equal to 4 ... etc. For the section angle θ, the *weights* are set as $W_{ij} = w_i$, w_i $cos(\theta/n)$, w_i, $w_i cos(\theta/n)$,.....(repeat w_i, $w_i cos(\theta/n)$ with total n+1 terms). The knot vector in direction I (s) is the same as the one of the generatrix while the other one in direction J (t) is determined according to the procedure described in the section of "Transform Circular Arc (entity 100) to NURBS curve". The control net and the *weights* are then transferred back to the original coordinates by reversing the translating /rotating operations. Figure 2.13 shows the construction of the associated control polygon at each cross section for the case of n equal to 2 (section angle equal to 90^0).

The final NURBS definition for the constructed surface in Figure 2.13 contains d_{ij} i=0,...,m, j=0,...,2 as the control net. The *order* and knot vector in the direction I are simply those of the generatrix, while the *order* in J direction

will be set as 3 and the knot vector is set as (0, 0, 0, 1, 1, 1). *Weights* are $W_{ij} =$
$(w_i,\ w_i cos(90/2),\ w_i)$ $i=0,..,m$ for all j.
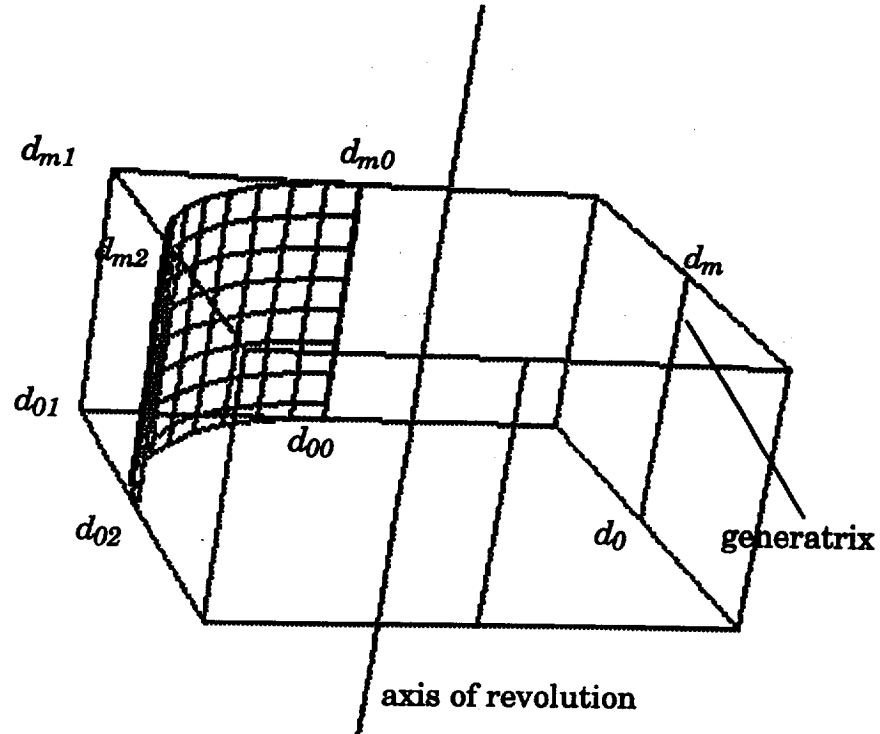
Figure 2.13 Illustration of surface of revolution by NURBS construction.

Figure 2.14 illustrates an example for this algorithm. This figure displays the "candle stand" NURBS control nets as well as the revolved surfaces for different starting and ending angles. Note the left figure also shows the generatrix.
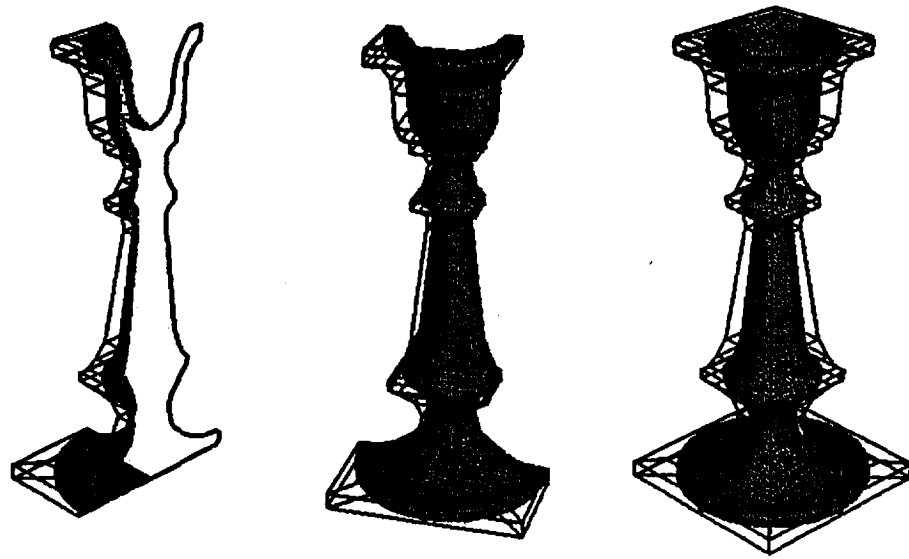
Figure 2.14 NURBS surfaces represent different surface of revolution.

Transform Tabulated Cylinder (Entity 122) to NURBS Surface

IGES defines the Tabulated Cylinder (entity 122), or extruded surface, as a surface formed by moving a line segment (called generatrix) parallel to itself along a curve (called directrix). In other words, given a NURBS curve, one can generate another curve by extruding the given curve with a distance a along a vector V. Then the control polygon of this new curve is determined by $\bar{d}_i = d_i + aV$. This NURBS tabulated cylinder is then defined as equation (2.15):

$$S(s,t) = \frac{\displaystyle\sum_{i=0}^{m} \sum_{j=0}^{1} W(i,j)d(i,j)N_i^k(s)N_j^2(t)}{\displaystyle\sum_{i=0}^{m} \sum_{j=0}^{1} W(i,j)N_i^k(s)N_j^2(t)} \qquad (2.15)$$

The knot vector in I direction is the same at the knot vector of the given curve, while the knot vector in J direction is set to (0., 0., 1.0, 1.0). The *order k1* is the same as the *order* of given curve, while *k2* is set to 2. The *weights W(i,j)* are set to the *weights* of the two curves, and the control net is set to $d(i,0)=d_i$ and $d(i,1)=\bar{d}_i$ for $i=0, ..., m$. Figure 2.15 shows this algorithm for a tabulated cylinder.
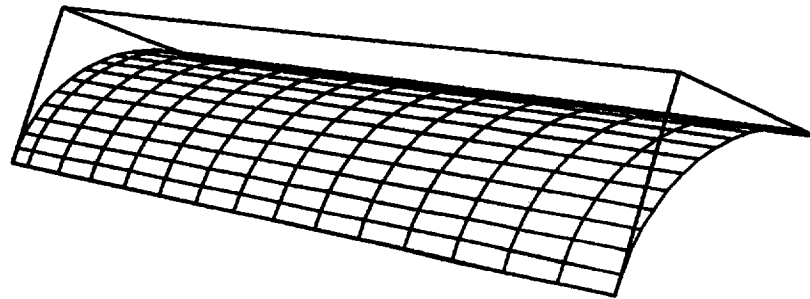
Figure 2.15 NURBS surface represents the tabulated cylinder.

Remark

In this chapter, the formulas of NURBS are first discussed. The transforming algorithms for converting different curve and surface entities to NURBS are presented. In the IGES standard, there are several other entities which are not discussed in this chapter. The reason of that is many of the entities are not related to curve and surface definitions. For example, the entity 106 (copious data entity), entity 116 (point entity) and entity 132 (connect point entity) will be treated as points, and the entity 124 (transformation matrix entity) is used for geometric transformation. And the other reason is the entities are already defined as NURBS, such as entity 126 (NURBS curve) and entity 128 (NURBS surface). Some entities will be discussed in later chapters. For example, the entity 130 (offset curve) and entity 140 (offset surface) will be discussed in Chapter Three. Two entities — entity 142 (curve on a parametric surface entity), entity 144 (trimmed parametric surface) will be discussed in Chapter Seven.

CHAPTER III

NURBS IN STATIC GRID GENERATION

Utilizing the NURBS definitions for generating grids is the focus of this chapter. The organization of this chapter is as follows. First, the basic concept of surface grid generation is introduced. Utilizing the "parametric" property of NURBS entity to generate different topologies for the same surface structure is mentioned in the next section. The third section covers the surface generation functions using NURBS for constructing the geometry which is commonly used in CFS analysis. The generating of volume grids by various NURBS options will be presented in the last section.

Basic Concept for Surface Grid Generation

Structured surface grid generation involves the establishment of a one—to—one correspondence between the non—uniformly distributed physical surface and the uniformly distributed computational plane. Let $\underline{r} = (x_1(s,t),$ $x_2(s,t), x_3(s,t))$ denote a parametric surface with Euclidean coordinates (x_1, x_2, x_3) having parametric values (s,t). The surface grid associated with the computational domain { $(\xi^1, \xi^2) \mid 1 \leq \xi^1 \leq n, 1 \leq \xi^2 \leq m$} will be denoted by $\underline{r} = ((x_1)_{ij},$ $(x_2)_{ij}$, $(x_3)_{ij})$, $i=1,..,n$ and $j=1,..,m$. Also there exists a one to one correspondence between the physical domain and the parametric distribution mesh. This one to one correspondence also exists between the surface distribution mesh and the computational domain. These relations are illustrated in Figure 3.1.

(x_1,x_2,x_3) (s,t) (ξ^1,ξ^2)

Physical space   Distribution mesh   Computation domain
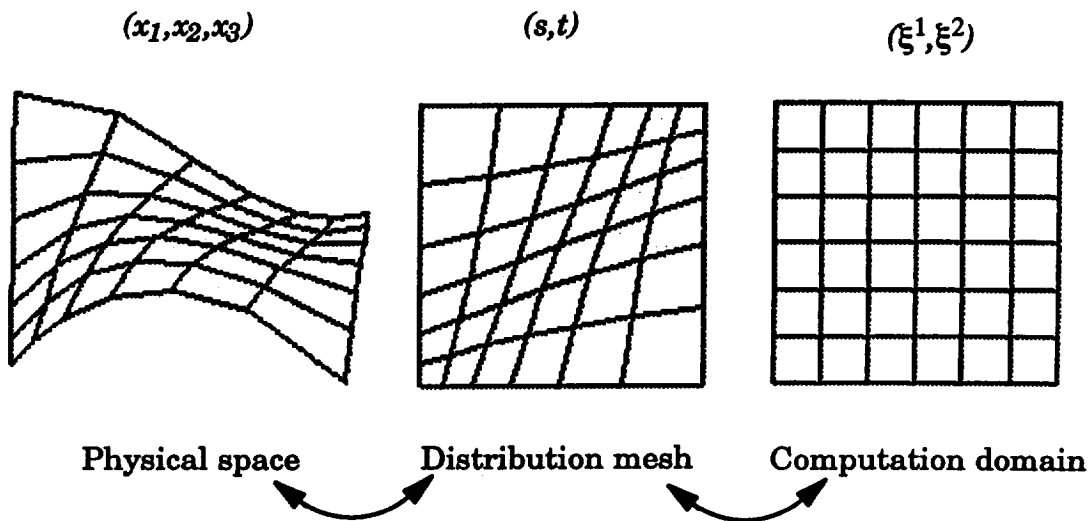
Figure 3.1 Physical space, Distribution mesh and Computation domain.

Parametric Grid Generation

From the NURBS formula listed in Chapter Two, it is easy to understand that the NURBS definition is presented in a parametric form. This means that for any parametric value t (s, t for the surface or s, t, u for the volume) in the parametric domain, there exists a unique point (excludes the singularity case) on the physical curve (or surface). This property actually can be applied to the other geometric definitions, such as the Bezier entity [Ref 23,24,25], cubic splines [Ref 23,34] and so on. However, the NURBS representation provides another unique property — the local control property which allows the designer to easily manipulate the geometry without altering the entire geometric definition. Utilizing these two properties, the NURBS can provide great flexibility in grid generation applications. For example, the generating of surface grids often requires different topologies — such as O type, C type and H type grid [Ref 72,73]. Generating a 3D surface from these different topologies for the same surface may require repeated applications of a routine. However, this goal can be achieved by using a 3D NURBS control polygon

(control net for the surface or control volume for volume case) and *weights* while generating the distribution mesh in 2D parametric space. This procedure is also referred as "griding in parametric space" [Ref 93]. After generating the two dimensional distribution mesh $(s(\xi^1,\xi^2), t(\xi^1,\xi^2))$, one can evaluate the NURBS entity by utilizing these 2D grid points as the parametric values to achieve the final NURBS entity. The advantage of doing this is that it is always easier to generate the grid in 2D parametric space within the [0,1] square than as a complex 3D surface. Even though this approach may result in an unexpected grid lines in physical NURBS entity, yet this problem can be overcome by the "re–parameterization" algorithm which will be described in next chapter. Different topologies of NURBS surface grids generated by using the same NURBS control points and *weights* are demonstrated in Figure (3.2) ~ (3.4).
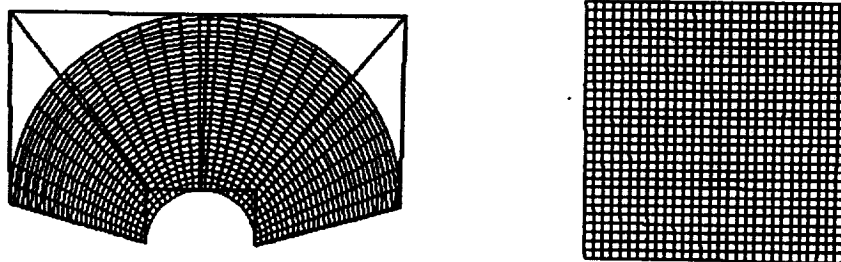
Figure 3.2 A *O*–type NURBS surface grid and the parametric values.
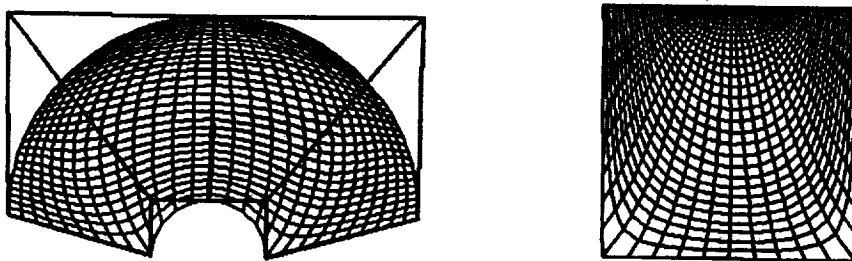
Figure 3.3 A *H*–type NURBS surface grid and the parametric values.
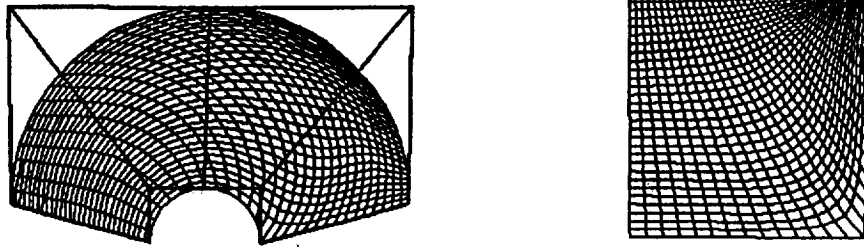
Figure 3.4 A *C*–type NURBS surface grid and the parametric values.

Here, the change in topologies of physical surface grid orientation is accomplished by changing the topologies of the distribution mesh. The same strategy can also be applied to the unstructured and hybrid cases. Figure (3.5) demonstrates the unstructured approach, and Figure (3.6) shows two 3D surfaces created from the hybrid approach.
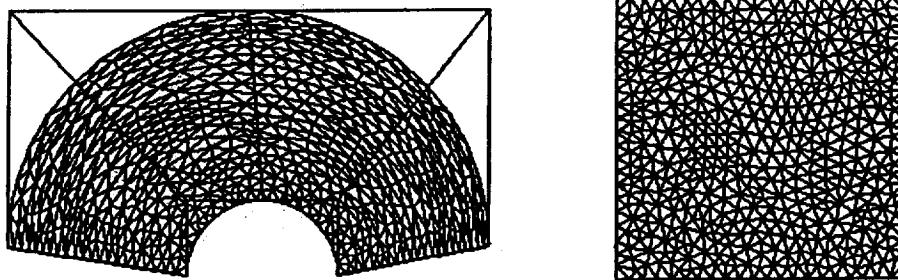
Figure 3.5 A unstructured NURBS surface grid and the parametric values.
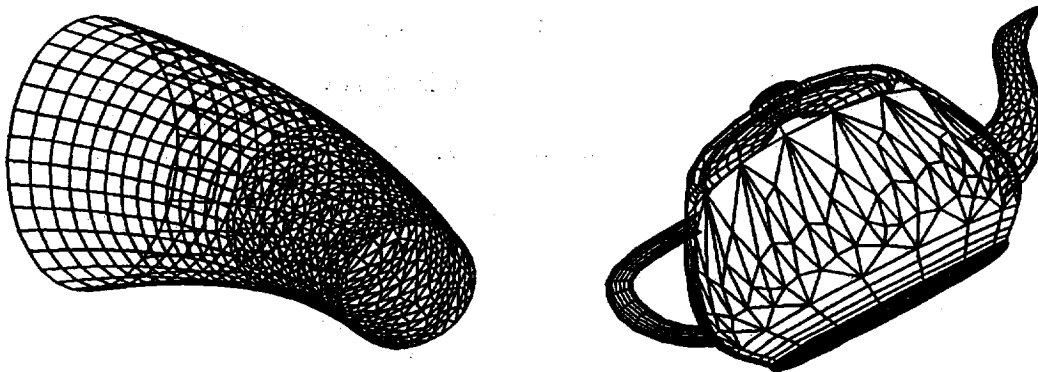
Figure 3.6 Two NURBS hybrid surface grids.

Surface Grid Generation by NURBS Control Net

The transforming procedures described in Chapter Two are used to transform an existing geometric definition from one form to another. For example, one can transform an existing cubic parametric surface to a NURBS definition. However, the generating procedures described in this chapter need to create the geometry definition from scratch. Two NURBS surface generation functions which are frequently used for creating the grid for CFD analysis are presented.

Transfinite Interpolation for NURBS Surface

From the examples shown in Figures 3.2 to 3.5, one can understand that different topology grids can be generated by different types of parametric values. This goal can also be achieved by constructing different types of control polygons and evaluating with regular parametric meshes. Taking the Figure 3.3 as an example, in order to generate the H type grid, the intuitive way is to create the NURBS H type control net for the NURBS surface. In many of the numerical grid generation applications, the H type grid can be generated easily by the *transfinite interpolation* algorithm *(TFI)* [Ref 15,73]. As a matter of fact, the *TFI* algorithm is the most frequently used function for the grid generation.

TFI, also referred to as *Coons–Gordon* patch, is a bivariate interpolation constructed from the superposition of a set of univariate interpolation schemes by the formation of the *Boolean* sum projector [Ref 73]. In other words, given a set of boundaries (or isoparametric curves), the *TFI* is a function which constructs the interior surface grid bounded by the given boundaries. The *Boolean* sum operator for a surface is defined in equation (3.1).

$$PS = P_\xi \oplus P_\eta = P_\xi(S) + P_\eta(S) - P_\xi P_\eta(S) \qquad (3.1)$$

where $P_\xi(S)$ interpolates the ξ direction of boundaries (the given isoparametric curves) and $P_\eta(S)$ interpolates the η direction of boundaries, while $P_\xi P_\eta(S)$ captures the failures of $P_\xi(S)$ and $P_\eta(S)$. The final surface PS bi–directionally interpolates the given boundaries / isoparametric curves and forms interior surface grid points bounded with the given curves. There are many functions which can be applied to *TFI*. For example, one can use the linear, quadratic or even cubic interpolation for function P in equation (3.1). Taking the linear interpolation for a surface with the resolution N by M as an example, the equation (3.1) can be re–written as equation (3.2).

$$R_{i,j} = (1 - s_{i,j})\, R_{i,j} + s_{i,j}\, R_{N,j} + (1 - t_{i,j})\, R_{i,1} + t_{i,j}\, R_{i,M} - \tag{3.2}$$
$$\left((1 - s_{ij})(1 - t_{ij})R_{11} + (1 - s_{ij})t_{ij}R_{1M} + s_{ij}(1 - t_{ij})R_{N1} + s_{ij}t_{ij}R_{NM}\right)$$

Variables R_{ij} in equation (3.2) are the control vertices which need to be determined. For the NURBS case, the R_{ij} could be d_x, d_y, d_z (control points) and w_{ij} (the *weights*).

This *TFI* function is a fundamental tool for generating grids in many grid applications. However, equations (3.1) and (3.2) can not be applied to NURBS *TFI* directly. The reason for this can be explained as follows: when four NURBS curves are given to generate a NURBS *TFI* surface, the interior control points can be created according to equation (3.2) (for the bi–linear interpolation) by supplying the control vertices of the boundaries without a problem. The problem comes when determining the interior *weights*. The addition and subtraction operations in equation (3.2) may lead to the interior *weights* being negative or zero values. Any negative *weights* will destroy the convex hull property of a NURBS entity, while any zero *weights* will make the

control vertices lose their influence. This obstacle can be overcome by the modified NURBS *TFI* [Ref 37]. The formula for this is shown in equation (3.3).

$$\left|\begin{matrix} WP(S) \\ W \end{matrix}\right|_{ij} = \left|\begin{matrix} W_\xi W_\eta \left(P_\xi(S) + P_\eta(S) - P_\xi P_\eta(S) \right) \\ W_\xi W_\eta \end{matrix}\right|_{ij} \qquad (3.3)$$

Each term of the equation (3.3) (for the case of linear interpolation of P) is defined as follows: the $P_\xi(S)$ represents a NURBS "ruled surface" (described in previous chapter) with *weights* of W_ξ formed in ξ direction (hence, the *order* is 2, knot vector is [0, 0, 1, 1] in ξ direction), $P_\eta(S)$ represents another NURBS "ruled surface" with *weights* of W_η formed in η direction (*order* is 2, knot vector is [0, 0, 1, 1] in η direction) and the $P_\xi P_\eta(S)$ is a NURBS surface which is constructed by using the 4 corners points as the control net with *orders* 2 by 2 in ξ and η directions. This is demonstrated in Figure 3.7 and 3.8.
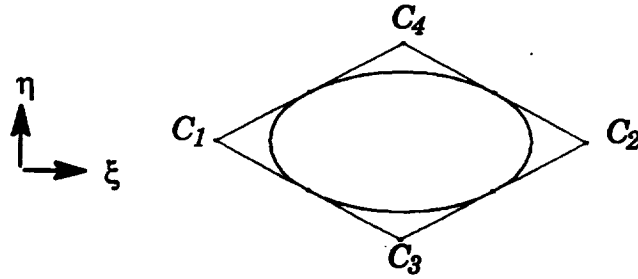
Figure 3.7 Four NURBS boundaries to form a *TFI* Surface.

After creating the intermediate surfaces of $P_\xi(S)$, $P_\eta(S)$ and $P_\xi P_\eta(S)$, one has to perform the "knot insertion" [Ref 8] and "degree elevation" [Ref 14] algorithms to these three surfaces to ensure all of them have the same *orders* and same knot vectors in both the ξ and η directions. If the NURBS surfaces have the same *orders* and same knot vectors, then the dimension of control net would be the same also. Therefore, the control net of the final NURB *TFI* sur-

face can be obtained by adding the control net of $P_\xi(S)$, $P_\eta(S)$ and subtracting those of $P_\xi P_\eta(S)$, while the *weights* are determined by multiplying W_ξ and W_η.
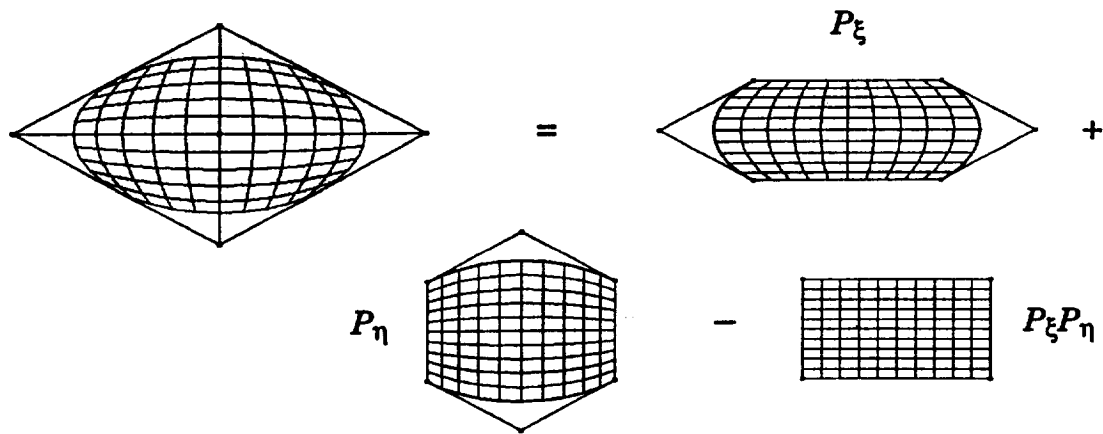
Figure 3.8 A superposition diagram for a NURB *TFI* surface.

Comparing the NURBS *TFI* with the traditional *TFI* shows that the NURBS *TFI* needs more computation because the *weights* need to be handled properly. In addition, the knot insertion and degree elevation algorithms need extra computation. However, this function is fundamental and useful when there is a need to create H type grids. Also, when generating the volume grids for a nozzle, this function is particularly useful to create the inlet and outlet surfaces. Figure 3.9 demonstrates this example.
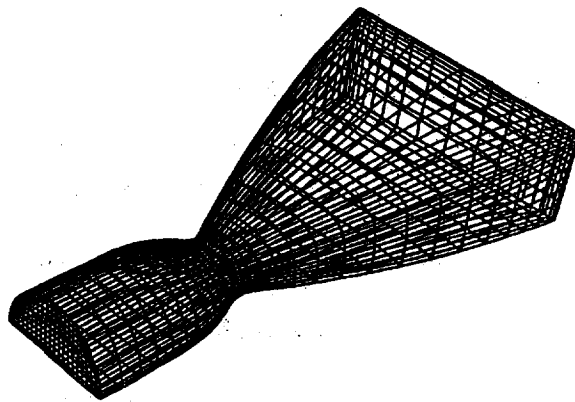
Figure 3.9 NURBS *TFI* creates the inlet / outlet surface for a nozzle.

Cascading Technique for NURBS Surface

As it is discussed in the transforming procedures (described in previous chapter), the surface of revolution algorithm can be used to model the symmetric surfaces. In CFD applications, some of the geometries for analysis are symmetric objects. An example of this could be the simulation of the flow passing around a missile. Generally speaking, the surface of revolution algorithm can be used to model a "simple" symmetric surface, but for many of the CFD applications, the real geometric objects interact with other objects and can not be modelled by rotating a boundary curve to form a surface of revolution. The example for this is shown in Figure 3.10 for a surface blade with the boundary intersected with a fin. Even though this surface is still symmetric, the surface of revolution (*SOR*) algorithm fails to model it.
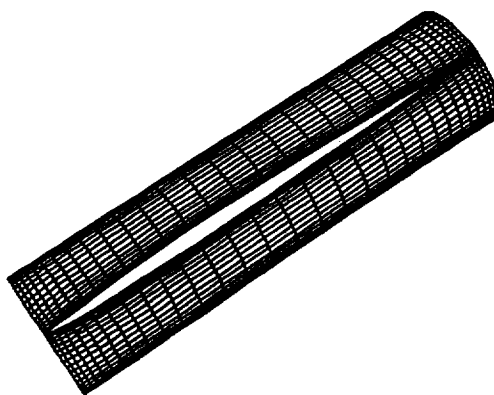
Figure 3.10 Example of symmetric surfaces couldn't be modelled by *SOR*.

This situation also happens to the "cascade" surface. A cascade surface is usually referred to the "blade–to–blade" surface in turbomachinery [Ref 56]. Even though most of the cascade surfaces are axis–symmetric, they can not be modeled by the NURBS surface revolution algorithm. Also, in the grid generation area, creating the surface grid for the cascade is a challenge. The diffi-

culty of modeling the surface grids for a 3D cascade surface is that when the blade leading edge (or trailing edge) circle radius is too big, such as the ones in a turbine, or if the blade setting angle (the blade angle) is very low, it is hard to generate a well behaved H type grid. Particularly, there is often a grid crossing near the leading edge (or the trailing edge) for such a geometry. Traditionally, this kind of surface grid is generated by transforming the 3D surface of the (x, y, z) coordinate to 2D parametric (m', σ) space, griding in the parametric space and then transforming them back to 3D physical space according to the relation of the (m', σ). Detailed information can be referred to in [Ref 53,54]. In this research, the NURBS modeling approach is presented for modeling this type of geometry.

The NURBS algorithm for modeling the cascade surface is described as follows: given a boundary curve of a cascade surface, transform it to a BSpline curve (as the curve A shown in Figure 3.11) by the interpolation technique (described in later section). Create a plane which bisects the surface sector angle (the θ, angle of ao_1b shown in Figure 3.11) and then use the "Mirror" function [Ref 50] to reflect curve A with this plane to create the curve C. The curve C will have the same $order$, knot vector and number of control points as those of curve A. The next step is to create a straight line lying on the plane which contains the points o_1, a and c. This is done by projecting the control polygons (described at the later section of this chapter) of curve A to this plane and setting the $order$, knot vector of this line to be the same as those of the curve A. After this line is created, perform the surface of revolution algorithm (refer to Chapter Two) to rotate this line with respect to the center of o_1o_2 for a total sector angle of θ. A NURBS tabulated cylinder (refer to Chapter 2) with sector angle θ will be generated after this step. However, this surface is not the de-

sired cascade surface. Hence, one has to perform the last step: replace the first and last iso–control polygons (in the axis direction) of this surface with the existing BSpline curves A and C. Because the tabulated cylinder is created by rotating a line which has the same *order* and knot vector as those of curve A, it is secure to replace the two control polygons of the surface with A and C without altering the entire shape of the surface. The control net, with curves A and C replacing the first and last iso–control polygons, is the final desired NURBS control net. A missile configuration, composed of the surface of revolution and cascade surface, is shown in Figure 3.12 to demonstrate this algorithm. Another example is shown in Figure 3.13 for a single rotation propfan modelled by this algorithm.
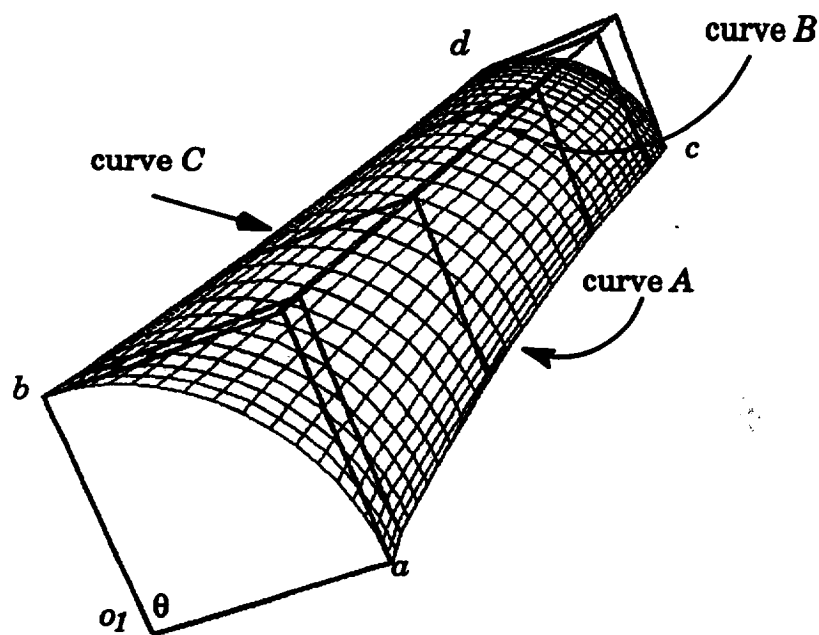
Figure 3.11 Illustration of modeling cascade surface by the NURBS.
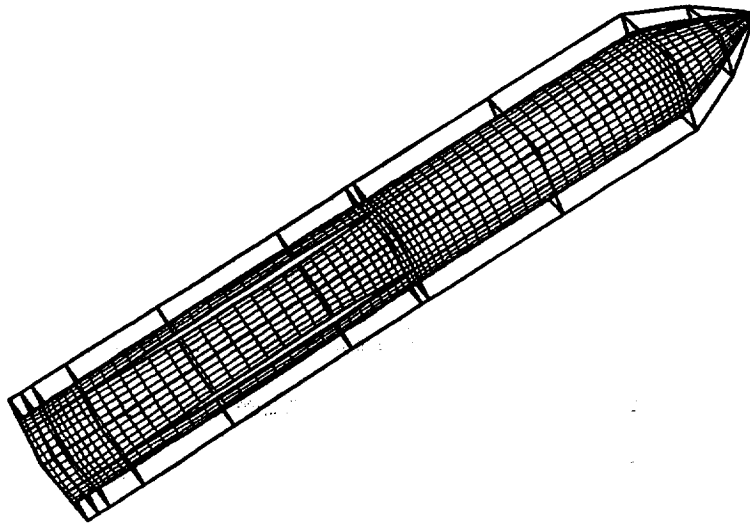
Figure 3.12 A missile configuration modeled by the NURBS.
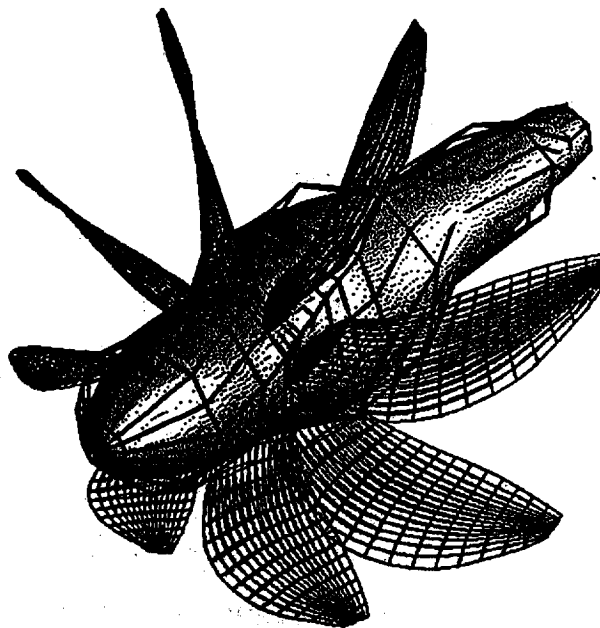
Figure 3.13 A single rotation propfan modeled by the NURBS.

Volume Grid Generation by NURBS Control Volume

Volume grid generation algorithms have been utilized in many CFD analysis procedures. A widely used technique to algebraically generate a three dimensional volume grid is to utilize the transfinite interpolation algorithm based on the bounding surface grids. However, the volume generation techniques are seldom applied to CAD/CAM applications. Even though NURBS representation has been widely used in many industry applications, the geometry modeled by the NURBS volume approaches are seldom discussed in CAGD (Computer Aided Geometry Design) literature. In this chapter, using the NURBS volume to model the geometry for the volume grid is presented. Observing the curve and surface examples shown in previous chapter, one can realize that modeling the geometry with NURBS may only need very concise control polygons. Instead of storing the surface grid points, one can store the associated control polygon (or control net for the surface) with the associated weights to reduce the memory load. This is especially useful for volume grid generation. Even though the computer memory availability has been dramatically improved, a complicated geometry usually consumes a lot of computer memory for the volume grid. Storing the NURBS control net to reduce the size of entire volume grid is demonstrated in the examples of this section.

The ultimate objective of the present research effort is to explore various NURBS control volume options applicable to three dimensional grid generation. In this chapter, the development of NURBS ruled volume, NURBS extruded volume, NURBS volume of revolution, NURBS composite volume and Transfinite Interpolation (*TFI*) NURBS volume are discussed.

NURBS Control Volume for a Ruled Volume

The easiest 3D NURBS volume to generate is the ruled NURBS volume. The algorithm is described as follows: Given two NURBS surfaces, the first step to form a ruled volume is making the knot vectors of the surfaces be in the same range of [0~1]. Next, considering the I direction of both surfaces, use the degree raising technique to raise the low degree ($order - 1$) of the surface. This procedure will yield a new knot vector and new control net. If the new knot vector differs from the other knot vector, then perform the knot insertion algorithm to merge them into one final knot vector. Then apply these steps to the knot vectors in J direction of both surfaces. After this step, the two NURBS surface will have the same *orders* and the same knot vectors in both I and J directions. This means the resolutions of control net of both surfaces will be the same. Finally, connect the corresponding control point together to form the 3D NURBS volume. The *orders* and knot vectors of the final volume in I and J directions will be the same as those of the surfaces after degree elevation and knot insertion, and the *order* in L direction will be set as two with knot vector set as (0,0,1,1). Figure 3.14 shows a 3D "apple–like" NURBS volume and its volume grid while Figure 3.15 shows a missile configuration with the control volume formed by this algorithm.

While defining the NURBS ruled surface, the IGES defines a variable named "*DIRFLG*" as a direction flag to control the direction of how the surface will be connected (refer to the previous chapter). If *DIRFLG* is 0, then the surface will connect the points in the same direction of the two curves, otherwise, the direction of one of the curves will be reversed. Similar to this definition, it is possible to set the two flags "*DIRFLG_I*" and "*DIRFLG_J*" to control how
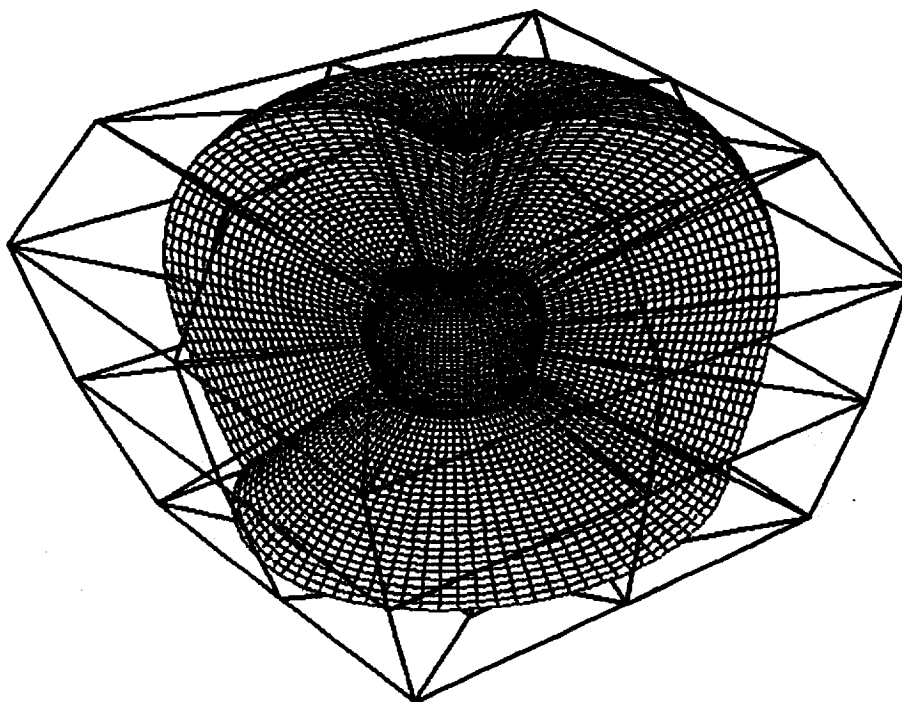
Figure 3.14 NURBS control volume and grids for ruled volume.
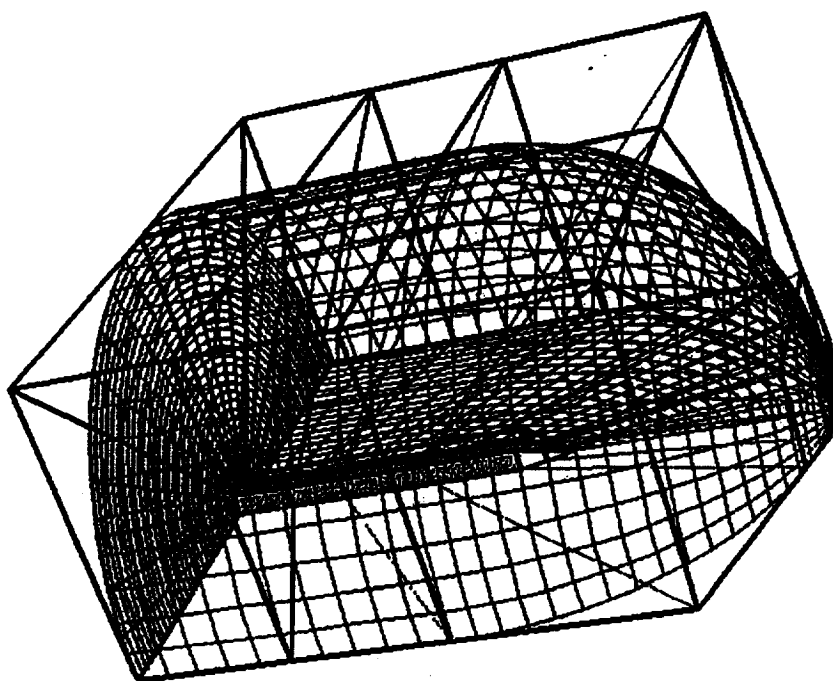
Figure 3.15 NURBS control volume and grids for a missile configuration.

the directions of the two surfaces will be connected. The use of these flags increases the flexibility of generation options.

NURBS Control Volume for an Extruded Volume

The generation of a NURBS extruded volume is an extension from the extruded surface definition. The IGES defines the extruded surface as a surface formed by moving a line segment parallel to itself along a curve. In other words, given a NURBS curve, one can generate another curve by extruding the given curve with a distance a along a vector V. Similar to this definition, the NURBS extruded volume is defined as follows: Given a NURBS surface with control net d_{ij} and the associate *weights*, knot vectors and *orders*. The new surface $\overline{d}_{ij}$ can be generated by "extruding" the given surface with a distance a along a vector V. Mathematically, this new NURBS extruded surface can be described as $\overline{d}_{ij} = d_{ij} + aV$ with the same *orders*, same *weights* and same knot vectors as those of the given surface. After this step, the algorithm of "ruled volume" (described in previous section, it simply connects these two surfaces and sets the L direction linearly) can be applied to these surfaces to form a final NURBS volume. Figure 3.16 shows an example of a 3D NURBS extruded volume.
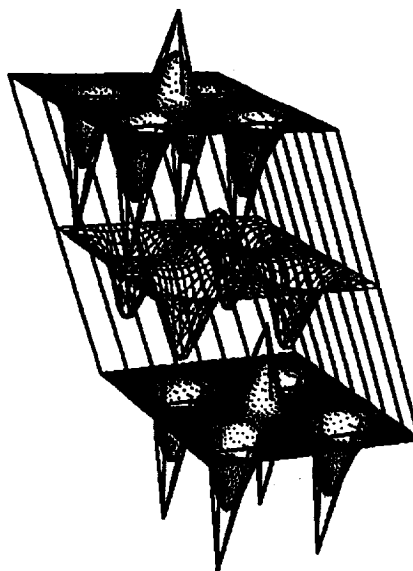
Figure 3.16 NURBS control volume and grids for extruded volume.

NURBS Control Volume for a Volume of Revolution

Another commonly used approach for generating volume grids is the "revolution" method. A revolution resulting in a surface is known as a "surface of revolution", while a revolution resulting in a volume is then defined as a "volume of revolution". The fact that this modelling technique can only be used for symmetric geometries is not limiting, as many objects in real world applications, such as turbomachinery configurations, are symmetric. The extension of a volume revolution modeled by NURBS is presented as follows: the definition of surface of revolution is a surface which is formed by rotating a given curve with respect to an arbitrary straight line from a starting angle to an ending angle (referred to Chapter 2). Likewise, the volume of revolution is defined as a volume form by rotating a given NURBS surface with respect to an arbitrary axis of revolution from any starting angle to an ending angle.

The general algorithm is outlined as follows: the first step is translating / rotating the axis of revolution by proper transformation matrix so that it is

coincident with the Z axis. This transformation matrix is also applied to the given NURBS surface so that the entire surface can be kept in the same position as the axis of rotation. It is assumed that the surface is defined (or transformed) as NURBS with the control net d_{ij}, *order k1* and *k2*, *weights* W_{ij} and two knot vectors. The second step is to construct, for each control net d_{ij} (on the generatrix $i= 0,..,m$, $j= 0,..,n$), the control volume d_{ijl} $l=0 ... p$ at each j-th cross section through the starting and ending angle by utilizing the circular arc algorithm. In other words, this approach constructs the NURBS control net at each J constant plane by revolving the control polygon d_{iJ} with respect to L direction and then "stack" them together to form a final NURBS volume. Figure 3.17 demonstrates this approach. The general procedure of generating the NURBS circular arc is described in Chapter 2. The p for the last dimension of control volume is determined by the sector angle θ (equal to the difference between ending and starting angle). For example, if the angle is less than 90^0, p is equal to 2. If the angle is in the range of $90^0 \sim 180^0$, p is equal to 4, if in the range of $180^0 \sim 270^0$, p is 6, if it is greater than 270^0, p should be 8. For the sector angle θ, the *weights* are set as (in each J constant plane, $J=0,..n$)

$W_{iJp} = w_{iJ}, w_{iJ} cos(q/p), w_{iJ}, w_{iJ}cos(q/p),.... \quad i=0,..,m$ (repeat $w_{iJ}, w_{iJ}cos(q/p)$

with total $p+1$ terms). The knot vectors in directions of I (s) and J (t) are the same as the ones of the given surface while the knot vector in direction L (u) is determined according to the circular arc procedure. For example, when p is 2, the associated knot vector is set as (0, 0, 0, 1, 1, 1), for the case of p equal to 4, the knot vector is set as (0, 0, 0, 1/2, 1/2, 1, 1, 1), for the case of p equal to 6, the knot vector is set as (0, 0, 0, 1/3, 1/3, 2/3, 2/3, 1,1,1) and for the case of p equal to 8, the knot vector is (0, 0, 0, .25, .25, .5, .5, .75 , .75 ,1,1,1). Also, set the

orders in I and J be *k1* and *k2* (as the *orders* of the original surface) while set 3 as the *order* of L direction.
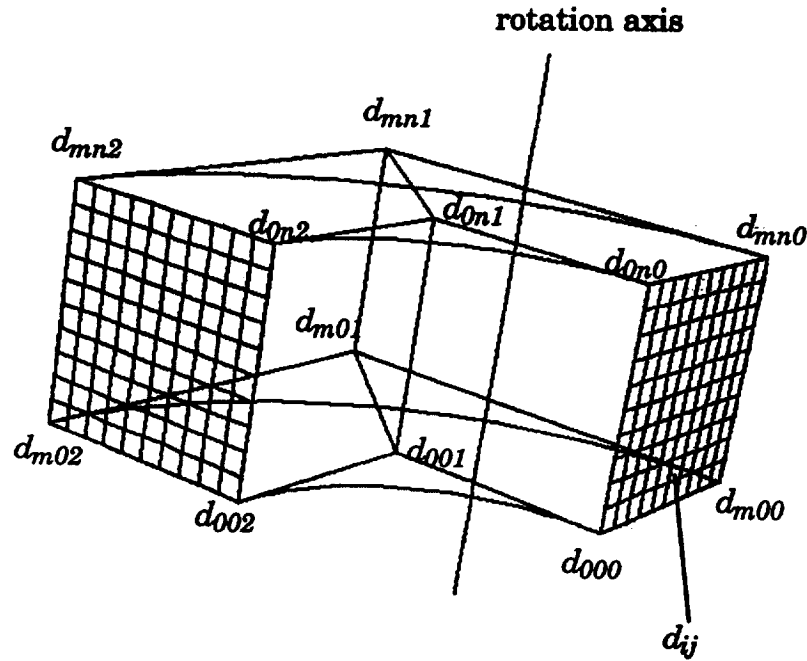
Figure 3.17 Illustration of constructing the NURBS volume.

Because the NURBS has the translate and rotate invariant properties, the inverse transformation matrix can be applied to the control volume (without altering the *weights* and knot vectors) returning the volume to the original coordinates. Figure 3.18 shows a 3D volume grids and its control volume according to this algorithm. This example was developed by revolving the *TFI* surface from 0^0 to 180^0. Because the NURBS surface *TFI* technique needs four boundary curves to define a surface, this will result in a "*H*" type surface grids. Revolving this "*H*" type *TFI* surface creates "*H*" type NURBS control volume and yields the "*H*" volume grids. This topology can be changed by revolving another "*O*" type NURBS surface to form an "*O*" type volume grids as shown in Figure 3.19. Notice that the sizes of this control volume are only

3x3x5 (for "*H*" type grids) and 9x9x5 (for "*O*" type grids), yet the resolution of the entire volume grid could be any number (for this case, 31 by 31 by 61).
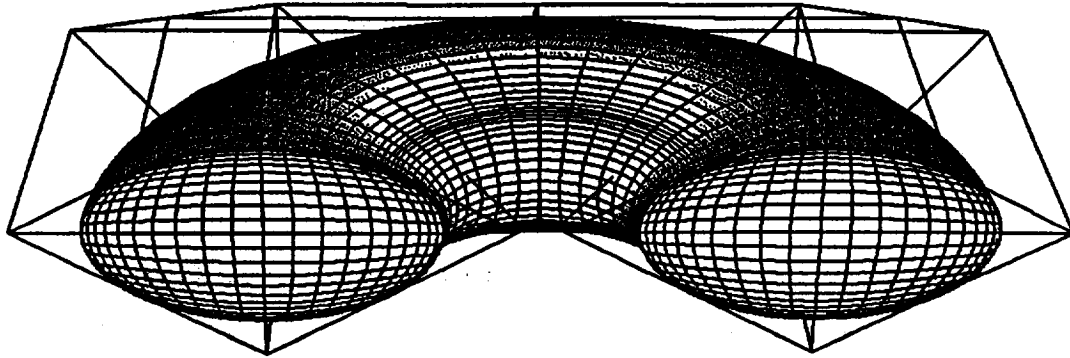
Figure 3.18 NURBS revolution volume for *H* type volume grid.
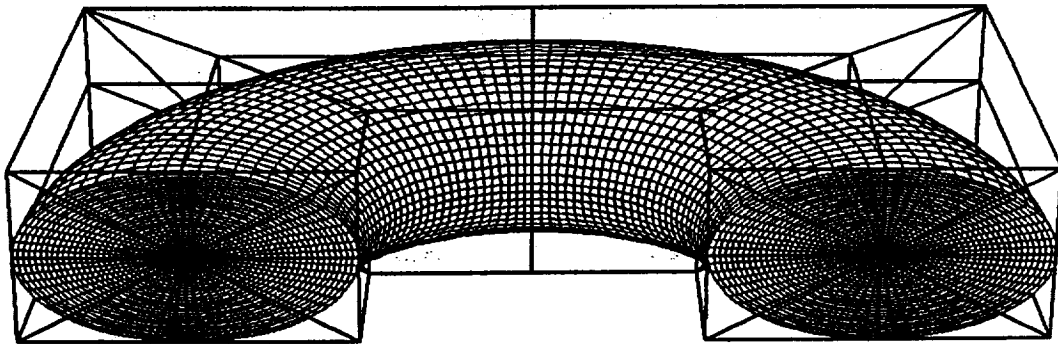
Figure 3.19 NURBS revolution volume for *O* type volume grid.

NURBS Control Volume for a Composite Volume

A composite NURBS volume is defined as a volume consisting of lists of constituent volumes. The composing procedure is stated as follows: Suppose two constituent NURBS volumes V_1 and V_2 form a composite volume. Assume that V_1 has control volume $d_1[0:m_1,\ 0:n_1,\ 0:l_1]$, weight $W_1[0:m_1,\ 0:n_1,\ 0:l_1]$, three knot vectors $knot\_i)_1$, $knot\_j)_1$, $knot\_l)_1$ and orders $k\_i)_1$, $k\_j)_1$, $k\_l)_1$ while V_2 has control volume $d_2[0:m_2,\ 0:n_2,\ 0:l_2]$, weight $W_2[0:m_2,\ 0:n_2,\ 0:l_2]$,

three knot vectors $knot\_i)_2$, $knot\_j)_2$, $knot\_l)_2$ and orders $k\_i)_2$, $k\_j)_2$, $k\_l)_2$. There are many possible combinations of the two volumes joined together. For example, one may join the volumes in I direction with the interface of J, L surface, or join in L direction with the interface of I, J surface, etc. Even though there are many cases, the procedure is similar. Take the case when joining in I direction as an example, the first step is to perform the degree elevation to V_1 and V_2 so that these two volumes can have compatible degrees in I, J and L directions. If the two knot vectors in J direction for V_1 and V_2 are not the same, merge them together by setting the final knot vector as { $knot\_j)_1$ | $knot\_j)_2$ }, then apply the knot insertion to V_1 and V_2 in J dimension. The same procedure should be applied to L direction if $knot\_l)_1$ and $knot\_l)_2$ are not the same. After this step, V_1 and V_2 will have the same degree in three directions, and the number of control points and knot vectors in J and L directions will be the same. The second step is to adjust the knot vector $knot\_i)_2$ so that its first knot value can be the same as the last knot value $knot\_i)_1$. Shifting the knot vector will not change the original NURBS because the basis function is a "normalized" basis function. The third step is to build up the final knot vector by joining the two knot vectors into one knot vector and setting that knot value at the joint point to have the multiplicity equal to (order −1). For example, if the knot vector $knot\_i)_1$ is [0., 0., 0., 1., 1., 1.] and the knot vector $knot\_i)_2$ is [2., 2., 2., 3., 3., 3.], adjust the second knot vector by shifting −1 to each value. Thus, the $knot\_i)_2$ becomes [1,1,1,2,2,2]. Suppose the final *order* of these two volumes in I direction is three, then, the final knot vector should be [0., 0., 0., 1., 1., 2., 2., 2.] (notice the interior knot 1 has multiplicity of (order−1)=2). The fourth step is to match the *weights* at the interface surface by multiplying the ratio of $W_1[m_1, j, l]$ / $W_2[0, j, l]$ to $W_2[i, j, l]$ for $i=0, \ldots, m2,$

j=0, ... , n1 and *l=0, ... , l1*. The last step is to construct the final control volume and *weights* by removing the $d_2[0{:}0,\ 0{:}n_2,\ 0{:}l_2]$ and $W_2[0{:}0,\ 0{:}n_2,\ 0{:}l_2]$ and joining the others as one control volume and weights.

Figure 3.20 demonstrates this algorithm. This 3D NURBS composite volume can be easily created by the following steps: First, use the algorithm of generating the ruled surface to form a NURBS surface by connecting two full NURBS circles, then extrude this surface with a proper distance α to form the first sub volume (the cylinder pipe) by the extruded algorithm. Second, extract the last cross section surface and revolve it with 90^0 to form the second sub volume (the turning portion) by the revolving algorithm. The third step is exacting the last cross section surface of the second sub volume and revolving it 180^0 to form the third sub volume (the turning portion). The forth step is extracting the last cross section surface of the previous volume and extruding it with a proper distance α to form the last sub volume (the cylinder pipe) by the extruded algorithm, and the last step is applying the composite algorithm to form the final volume. The dimension of this NURBS control volume is only 9 by 11 by 2, yet it can create volume grids as large as 121 by 91 by 31. The flow field solution obtained using the NPARC [Ref 16] flow simulation code for this configuration is also included.
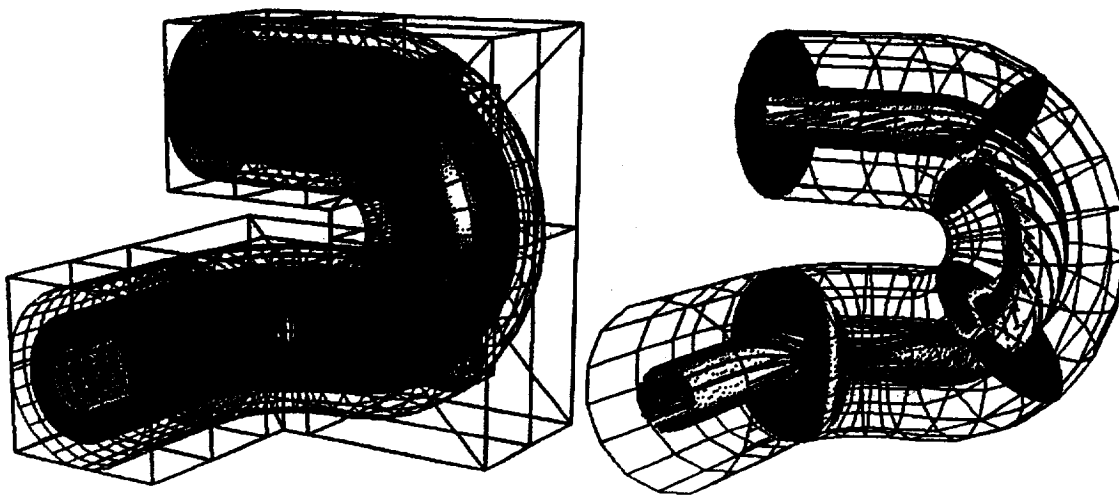
Figure 3.20 A composite NURBS pipe volume with the NPARC solution.

Generally speaking, it is difficult to model a complicated geometry by a single NURBS control volume. However, one can construct the individual control volume and then utilizing this composite algorithm to merge for a final volume. Figures 3.21 and 3.22 demonstrate the flexibility and the advantage of this approach. The NURBS control volumes are used to model the internal pipes. The griding of the turning portions of those circular pipes is a time consuming and tedious task in grid generation. However, the turning portions can be constructed by "volume of revolution" without any difficulties. Assembling all the sub NURBS volumes makes the final single block NURBS control volume.
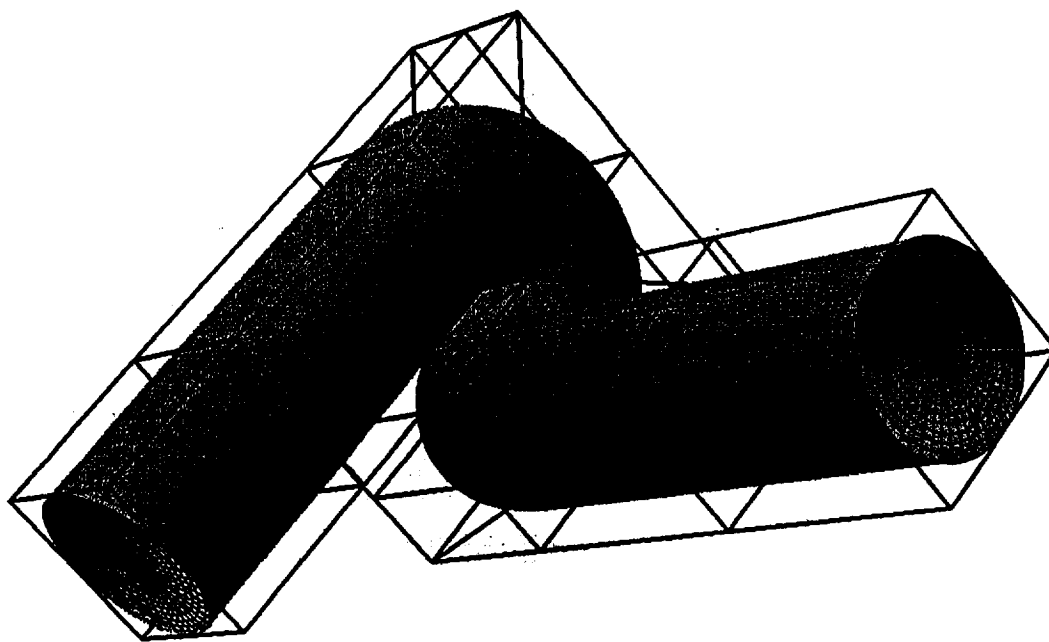
Figure 3.21 Composite NURBS volume for a turning pipe (i).
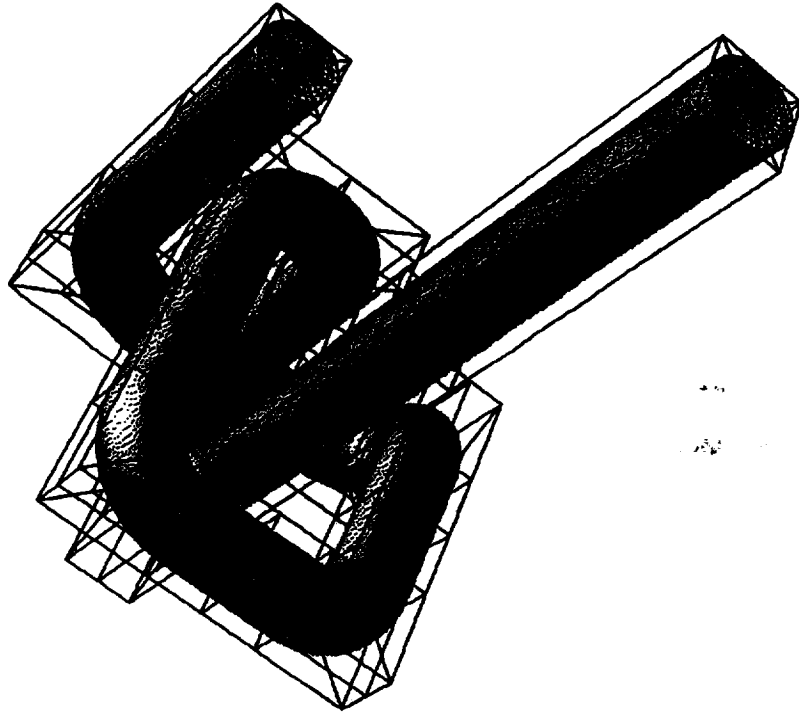
Figure 3.22 Composite NURBS volume for a turning pipe (ii).

NURBS Control Volume for a *TFI* Volume

Similar to the NURBS *TFI* surface algorithm, this approach is frequently used to generate an H type volume grid. Instead of providing 4 NURBS curves, this algorithm requires six NURBS surfaces to generate a NURBS *TFI* volume. This algorithm is the extension from the surface to volume, hence, the *Boolean* sum equation is defined as equation (3.4).

$$PV = P_\xi \oplus P_\eta \oplus P_\zeta =$$
$$P_\xi V + P_\eta V + P_\zeta V - P_\xi P_\eta V - P_\eta P_\zeta V - P_\xi P_\zeta V + P_\xi P_\eta P_\zeta V \qquad (3.4)$$

The P could be any interpolation function, such as the linear, quadratic hermit or the cubic interpolation and so on. The traditional definitions of each term in equation (3.4) can be found in [Ref 73]. However, as it is dis-

cussed in the NURBS *TFI* surface section, the traditional *TFI* approach [Ref 61,73] can not be applied to the process of generating a NURBS *TFI* control volume because the addition and subtraction operations in equation (3.4) may lead to zero or negative *weights* in the interior control volume. Any zero *weight* will make the corresponding control point lose its influence and the negative *weights* will create undesirable grids, such as the un–bounded grids or crossing grids. Hence, when applying the equation (3.4) to a NURBS *TFI* volume, it is necessary to re–define the individual terms listed in equation (3.4). The procedure is shown as follows: suppose the six NURBS surfaces are all pre–defined, and the surfaces of S_1 and S_2 are used for the ξ direction. S_1 and S_2 have the same *orders* of *k2, k3*, and same number of control points of n by L (refer to Chapter Two). If the *orders* of these two surfaces don't match, one should perform the degree raising algorithm to the low degree surface. If the resolutions of the control points of S_1 and S_2 are different, then the knot insertion algorithm should be used to make them the same. The same proce-dures should be applied to the surface of S_3, S_4 (with the *orders* of *k1, k3* and the resolutions of control net of m by L) and S_5, S_6 (with the *orders* of *k1, k2* and the resolutions of control net of m by n). After this step, the definition of each term for a linear NURBS *TFI* volume can be defined as follows: the $P_\xi V$ is a NURBS volume which is created by using the surfaces of S_1 and S_2 with the algorithm of the "Ruled NURBS volume" (described in the previous section of this chapter). Hence, the three *orders* of $P_\xi V$ are *2, k2* and *k3*, while the reso-lutions of the control volume is *2* by n by L. The same procedures should be applied to $P_\eta V$ and $P_\zeta V$. Therefore, the *orders* of $P_\eta V$ are *k1, 2, k3* with the resolutions of the control volume of m by *2* by L, while the *orders* of $P_\zeta V$ are *k1, k2, 2* with the resolutions of the control volume of m by n by *2*. $P_\xi P_\eta(V)$ is a

NURBS volume which is created by utilizing the boundaries (in ζ direction) of S_1, S_2 and the corner points of S_3, S_4, S_5, S_6. In other words, it has *orders* of 2, 2, *k3* and the dimension of control volume of 2, 2, *L*. The $P_\eta P_\zeta(V)$ and $P_\xi P_\zeta(V)$ are defined analogous — the *orders* of the $P_\eta P_\zeta(V)$ are *k1*, 2, 2 and the resolution of the control volume is *m*, 2, 2, while the *orders* of the $P_\xi P_\zeta(V)$ are 2, *k2*, 2 and the resolution of the control volume is 2, *n*, 2. The last term of $P_\xi P_\eta P_\zeta(V)$ is simply a NURBS control volume constructed by all the corner points of the six surfaces. Hence, the *orders* of this volume are 2, 2, 2 and the size of control volume is 2 by 2 by 2. These seven control volumes are illustrated in Figure 3.23.

After these seven intermediate control volumes are created, equation (3.5) should be used for the final linear NURBS *TFI*. This equation will avoid the creation of any undesired interior *weights*.

In addition to the algorithm of NURBS *TFI* surface, one has to perform the "knot insertion" and "degree elevation" [Ref 8,14] algorithms to all of these seven intermediate control volumes to ensure all of them have the same *orders* and same knot vectors in all the ξ, η and ζ directions, respectively. After this step is completed, the size of all the control volumes would be the same. Hence, the final control volume for NURB *TFI* can be obtained by adding the corresponding control points of $P_\xi(V)$, $P_\eta(V)$, $P_\zeta(V)$, $P_\xi P_\eta P_\zeta(V)$ and subtracting those of $P_\xi P_\eta(V)$, $P_\eta P_\zeta(V)$ and $P_\xi P_\zeta(V)$, while the *weights* are determined by multiplication of W_ξ, W_η and W_ζ. Figure 3.24 shows an *H* type nozzle generated according to this approach.
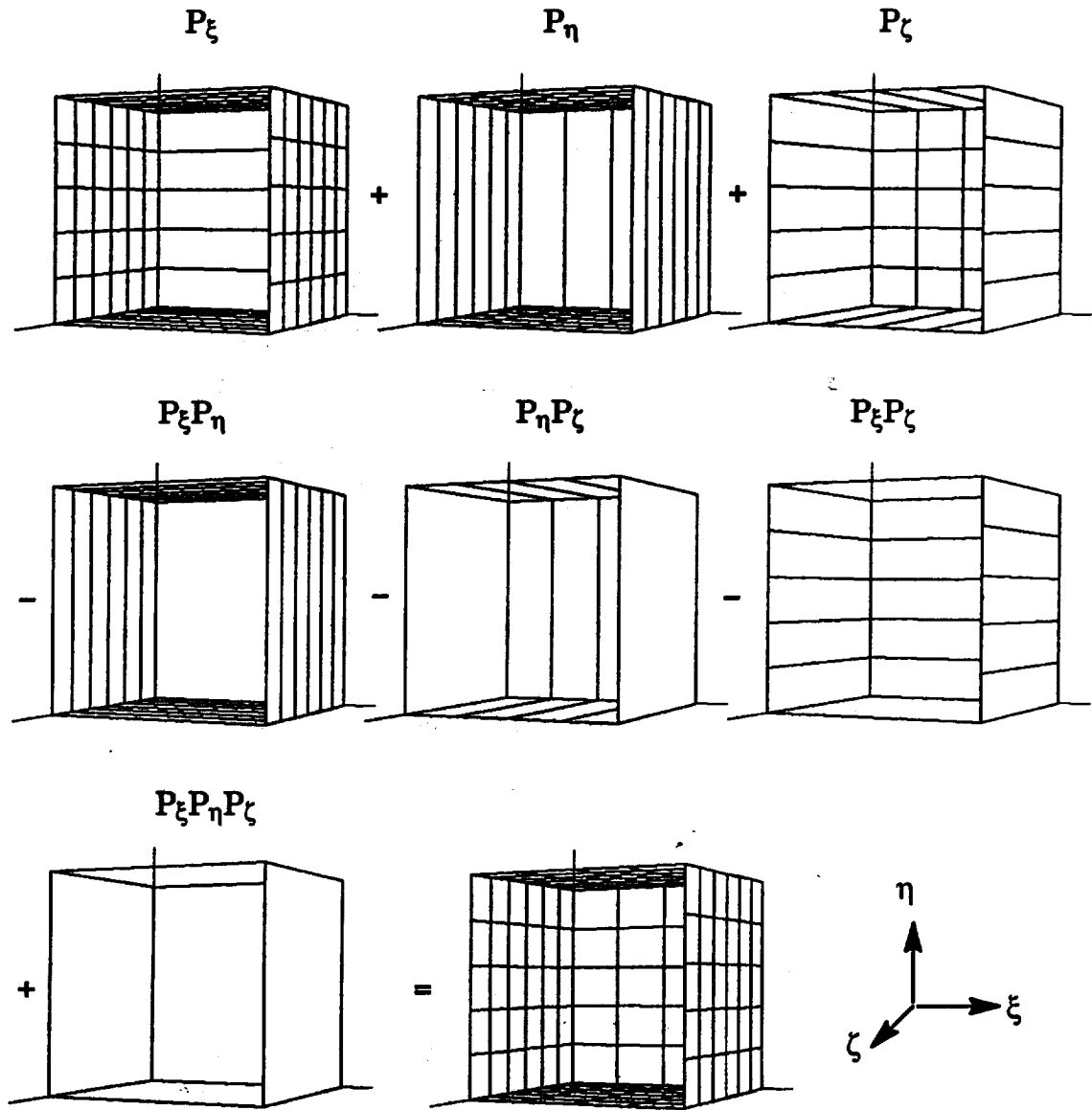
Figure 3.23 Illustration of NURBS TFI volume.

$$\left| \begin{matrix} WP \\ W \end{matrix} \right|_{ijk} = \left| \begin{matrix} W_\xi W_\eta W_\zeta \left(P_\xi + P_\eta + P_\zeta - P_\xi P_\eta - P_\eta P_\zeta + P_\xi P_\eta P_\zeta \right) \\ W_\xi W_\eta W_\zeta \end{matrix} \right|_{ijk} \quad (3.5)$$
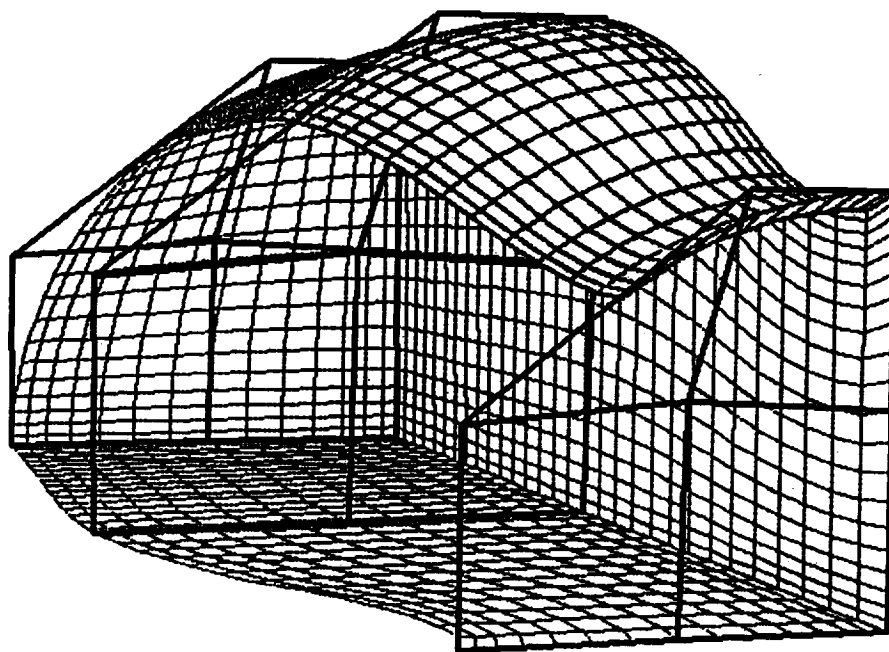
Figure 3.24 A NURBS TFI nozzle with H type volume grid.

General Interpolation Algorithm

Another category of generating grids is by the interpolation methods. This technique is frequently used for modeling the sculptured discretized data points. Many interpolating algorithms, such as the "cubic spline interpolation", "Lagrange interpolation", "BSpline interpolation" and "Hermite interpolation" are well known approaches. Among them, the BSpline interpolation technique has been frequently utilized in many of CFD researches. The examples can be found in Yang's *General Purpose Adaptive Grid Generation System* [Ref 83,84,85] and Shih's *TIGER (Turbomachinery Grid Generation System)* [Ref 53,56]. Also, other grid generation softwares provide the interpolation option. These softwares include *Genie++*, *EagleView* and *NGP*. Many of the softwares use this technique only for certain *order* (most of them, *order* is set to four), In this study, a general BSpline interpolation algorithm which allows for an arbitrary *order* is presented.

The definition of interpolation for the curve data set is defined as follows: given discrete data points Q_j, $j=0,..,n$, construct the BSpline curve (set all the *weights* to 1 for the NURBS curve) such that the BSpline can pass through all the given data points Q_j. Mathematically, this is described in equation (3.6).

$$Q_j = C(t) = C(\overline{u}_j) = \sum_{i=0}^{n} d_i N_i^k(\overline{u}_j) \quad j = 0,...n \qquad (3.6)$$

From equation (3.6), there are four unknowns: d (control polygon), k (the *order*), $N_i^k(\overline{u}_j)$ (the basis function, which is related to the knot vector) and the parametric values $\overline{u}_j$. The $\overline{u}_j$ is used for evaluating the BSpline curve and can be obtained by the relative chord (arc) length parametrization or the "centripetal" parametrization [Ref 23,34] from the given data points Q_j. From the previous chapter, one knows that the normalized chord length parametrization will lead $\overline{u}_0 = 0.0$ and $\overline{u}_{n-1} = 1.0$. Then, the user has to decide what *order* (k) should be used (generally speaking, $k=3$ or 4 are sufficient). After the k and $\overline{u}_j$ are known, the knot vector T is determined as $T = [0, ... 0, t_1, ... , t_{n-k-1}, 1,...1]$ with the knot value t_j defined as equation (3.7).

$$t_j = \frac{1}{(k-1)} \sum_{i=j}^{j+(k-1)-1} \overline{u}_i \quad j = 1,..., n-(k-1) \qquad (3.7)$$

Once the information of k (the *order*), knot vector and $\overline{u}_j$ are all set, solving the equation (3.6) for the unknown d (the control points of Bspline) will result in a $(n+1)$ by $(n+1)$ linear equation. Notice that the knot values of 0 and 1 have the multiplicity of k, hence, the interpolated BSpline curve will have the first control point identical to the first of the data set ($d_0 = Q_0$), and the

last control point identical to the last point of the data set ($d_n = Q_n$). This linear equation is listed as equation (3.8).

$$
\begin{bmatrix}
1 & 0 & \cdots & 0 \\
N_0^k(\bar{u}_1) & N_1^k(\bar{u}_1) & \cdots & N_{n-1}^k(\bar{u}_1) \\
N_0^k(\bar{u}_2) & N_1^k(\bar{u}_2) & \cdots & N_{n-1}^k(\bar{u}_2) \\
\cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot \\
N_0^k(\bar{u}_{n-1}) & N_1^k(\bar{u}_{n-1}) & \cdots & N_{n-1}^k(\bar{u}_{n-1}) \\
0 & 0 & \cdots & 1
\end{bmatrix}
\begin{bmatrix}
d_0 \\ d_1 \\ d_2 \\ \cdot \\ \cdot \\ \cdot \\ d_{n-1} \\ d_n
\end{bmatrix}
=
\begin{bmatrix}
Q_0 \\ Q_1 \\ Q_2 \\ \cdot \\ \cdot \\ \cdot \\ Q_{n-1} \\ Q_n
\end{bmatrix}
\qquad (3.8)
$$

Based on the knowledge of NURBS, one knows that there are at most k non–zero basis functions for any particular $\bar{u}_i$ for *order* k. Hence, the system of equations in (3.8) are banded with width no larger than k. The *LU decomposition* or *Gauss elimination* [Ref 19] can be used to solve equation (3.8) for the control polygon d_i.

The interpolation of the BSpline surface for the discrete data set is analogous. It is defined as follows: given discrete data points Q_{ij}, $i=0,..,m$, $j=0,..,n$, construct a BSpline surface such that this surface can pass through all the given data points Q_{ij}. This can be presented as equation (3.9)

$$
Q_{ij} = S(\bar{u}, \bar{v}) = \sum_{i=0}^{m} \sum_{j=0}^{n} d_{ij} N_i^k(\bar{u}_i) N_j^k(\bar{v}_j) \quad i = 0,..m \; j = 0,..n \qquad (3.9)
$$

The $\bar{u}_i$ and $\bar{v}_j$ are the parametric values obtained by the relative chord (arc) length or the "centripetal" parametrization [Ref 23,34] from the given data points Q_{ij}. As is mentioned in previous section, the BSpline surface is

defined as a tensor product surface. Equation (3.9) can be rewritten to equation (3.10) shown as follows.

$$Q_{ij} = \sum_{i=0}^{m} \left[\sum_{j=0}^{n} d_{ij} N_j^k(\bar{v}_j) \right] N_i^k(\bar{u}_i) = \sum_{i=0}^{m} d_i^* N_i^k(\bar{u}_i) \qquad \begin{matrix} i=0, \ ..m \\ j=0, \ ..n \end{matrix} \qquad (3.10)$$

Therefore, the interpolation of a BSpline surface with any arbitrary *orders* can be implemented by interpolating each row of the control polygon by using equation (3.6) to obtain the intermediate cross section control polygons. Then each column of these intermediate control polygons must be interpolated to obtain the final control net.

A direct application of this interpolation is illustrated for the transforming algorithm for the IGES entity 130 (offset curve) and entity 140 (offset surface).

Transform Offset Curve (Entity 130) to NURBS Curve

An offset curve defined in IGES 5.1 is created by "offsetting" a existing parametric curve with a specified distance. The curve to be offset must be a planar curve, and must be slope continuous over the entire curve. Let this base curve be represented as $r(t)$, and let V be a unit vector normal to the plane which contains $r(t)$. Because $r(t)$ is a slope continuous planar curve, its unit tangent vector can be obtained. Let this tangent vector be $T(t)$, then the desired offset curve is defined as equation (3.11).

$$OC(t) = r(t) + f(s) \times (V \times T(t)) \qquad t_1 \leq t \leq t_2 \qquad (3.11)$$

Variables t_1 and t_2 are the starting and ending parametric values which must be chosen to be in the domain of the base curve $r(t)$. The $f(s)$ is a function controlled by a variable, *FLAG*, given in the IGES file. If the *FLAG* is equal to 1,

the equation (3.11) is a uniform offset curve. In other words, *f(s)* is set as a constant value, say D_1. The D_1 is used as the constant offset distance for the offset curve. If the *FLAG* is equal to 2, then the offset distance is set varying linearly from the distance of D_1 to D_2 by the equation (3.12).

$$f(s) = D_1 + (D_2 - D_1) \times \frac{(s - TD_1)}{(TD_2 - TD_1)} \qquad (3.12)$$

s, TD_1 and TD_2 are either the accumulated arc length or the parametric values defined in the IGES document.

After knowing how the offset curve *OC(t)* is created, it is easy to convert this offset curve to a BSpline representation. The first step is transforming the base curve to a NURBS representation according to the algorithms described in chapter 2. It is well known [Ref 3,4,31,33] that the *OC(t)* can not be generated by applying equation (3.11) to the NURBS control polygons of the base curve. The simplest reason for this is that most of the control polygons are not slope continuous. Also, the *order* of the curve is not preserved after the offset procedure. Hence, one should evaluate the base curve for the set of discrete data points with reasonable resolution. Furthermore, the tangent value should be stored at the vector *T(t)* at each data point. Then, equation (3.12) should be used obtain another set of points according to the variable *FLAG* (to have a uniform / non—uniform offset curve). The curve interpolation algorithm is then applied to these data points to obtain a BSpline curve with the desired *order*. Figure 3.25 shows the base curve (a NURBS curve with the control polygon) and a non—uniform offset curve obtained by this algorithm.
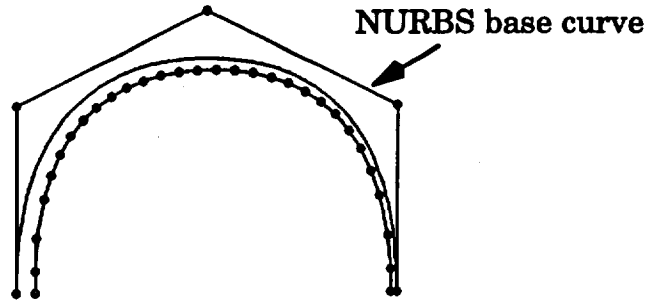
Figure 3.25 NURBS base curve and the non–uniform offset curve.

Transform Offset Surface (Entity 140) to NURBS Surface

The transforming procedure of this entity is analogous to the curve procedure. Given a base surface $S(u,v)$ which is defined by its specification parameters and oriented by $N(u,v)$, which is a differentiable field of unit normal vectors defined on the whole surface, the offset surface is then defined as equation (3.13), where D is a fixed nonzero real number used for the offset distance.

$$OS(u,v) = S(u,v) + D \times N(u,v) \quad u_1 \leq u \leq u_2 \; v_1 \leq v \leq v_2 \quad (3.13)$$

$N(u,v)$ is obtained by the cross product of the two surface derivatives. In other words, $N(u,v) = (\partial S/\partial u \times \partial S/\partial v) / \|\partial S/\partial u \times \partial S/\partial v\|$. Because this unit normal vector could have two directions, the IGES documentation also provides an "offset indicator" [Ref 35] to avoid confusion connecting the orientation of the base surface.

Transforming the offset surface to a BSpline surface is similar to the one of offset curve. The first step is to transform the base surface to a NURBS representation according to the algorithms described in Chapter Two. Again, it is not possible to apply the equation (3.13) to the control net of the base surface to form the offset surface. The correct procedure is to evaluate the base

surface for a set of data points with reasonable resolutions. Also, at each data point, the two surface derivatives should be calculated to obtain the union normal value $N(u,v)$. Then utilize the equation (3.13) to obtain the final set of points. The surface interpolation algorithm is then applied to these data points to obtain a BSpline surface (NURBS surface with all *weights* equal to 1) with the desired *orders*.

In these transforming procedures, one may have a question: Since the equations (3.11) and (3.13) can not be applied directly to the control polygon (or control net) of the base curve (surface), why is there a need to transform the base curve (or base surface) to the NURBS representation? The answer to this question is related to the accuracy of the tangent vector obtained from the base curve (or the union normal vectors from the base surface). Equations (3.11) and (3.13) require $T(t)$ and $N(u,v)$, which are the tangent vectors of the curve and the union normal vectors of the surface. Once the base curve and the base surface are transformed to the NURBS representation analytically (based on the transforming algorithms described on chapter 2), the variables of $T(t)$ and $N(u,v)$ can be calculated exactly. Using the exact derivative can reduce the interpolation error, and thus, increase the accuracy of the final offset curve / surface.

Geometry Modeling by Interpolation Technique

Besides the NURBS generation functions presented in previous sections, the interpolation technique is also a widely used approach for creating the curves or surfaces. It has been commonly used for transforming a discrete data set to a BSpline (the special case of NURBS) definition. Specifically, it has been used to fit the free–form or sculptured curves and surfaces. Unlike the approximation method, which does not require the final curves or surfaces

pass through the given data set, the interpolation method does require the curve or surface which must satisfy the given data set precisely.

From the equations (3.11) ~ (3.13), one can easily understand that there is a drawback for this technique — the disadvantage of this method is that the size of the control polygon of the interpolated BSpline curve will be the same as the size of the input data set. This destroys the advantage of saving the computer memory when using the NURBS definition for geometry modeling. when the size of data points are huge, this interpolated approach will result in a huge control polygon (or control net). Fortunately, the data reduction methods [Ref 5,30], which have been discussed in many literatures, can be used to reduce the redundant control points and maintain the desired accuracy. Figure 3.26 shows the BSpline curves which model an engine profile. The discrete data points are originally obtained from a digitizer. The curve interpolation method is then applied to these points to generate the BSpline control polygons. The associated control polygons have been reduced by the data reduction algorithm described in [Ref 10,38]. After defining these curves with BSpline representation, the surfaces for this axial symmetric engine can be generated by a simple rotation with respect to the X axis. Figure 3.27 and 3.28 show the associated surface grids and the NURBS control nets for this engine.
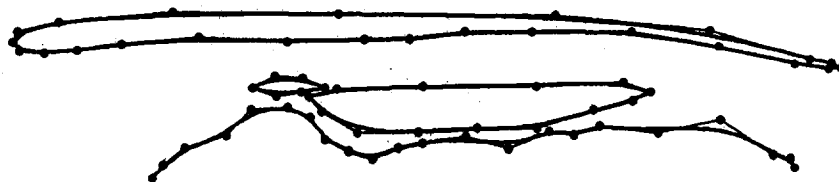
Figure 3.26 The curve interpolation technique for a engine profile.
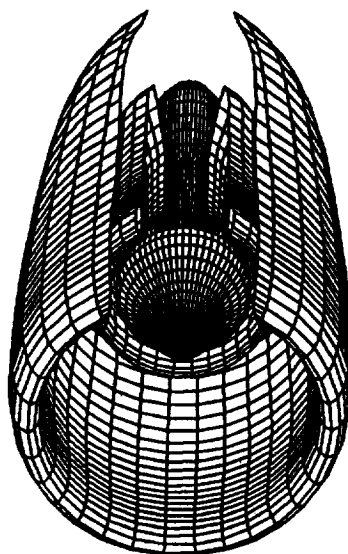
Figure 3.27 Surface grids for the multiple–duct engine.
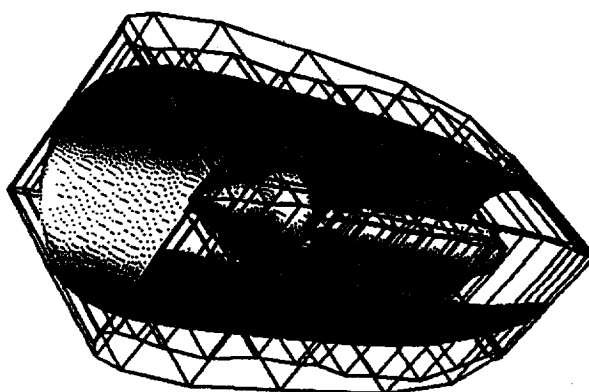
Figure 3.28 NURBS control nets for multiple–duct engine.

Modeling the volume grid for this geometry requires the multi–block strategy. A valid domain decomposition is to break the entire surface control nets into several NURBS *TFI* sub–patches and then apply the volume of revolution technique to these sub–patches to construct the entire multi–block NURBS control volumes. Figures 3.29 and 3.30 show the NURBS control volume and the volume grids.
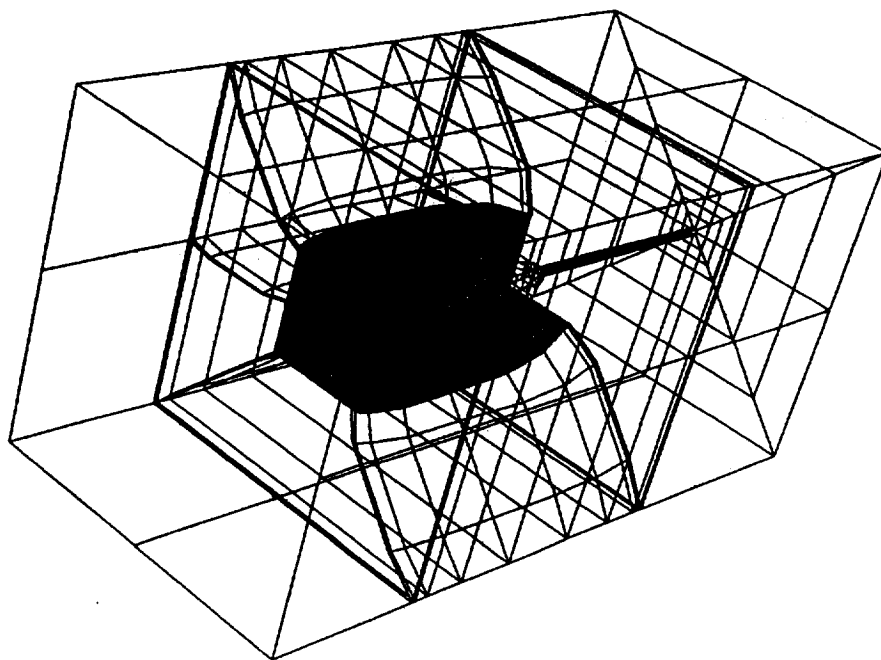
Figure 3.29 Multi–blocks NURBS control volume for mock engine.
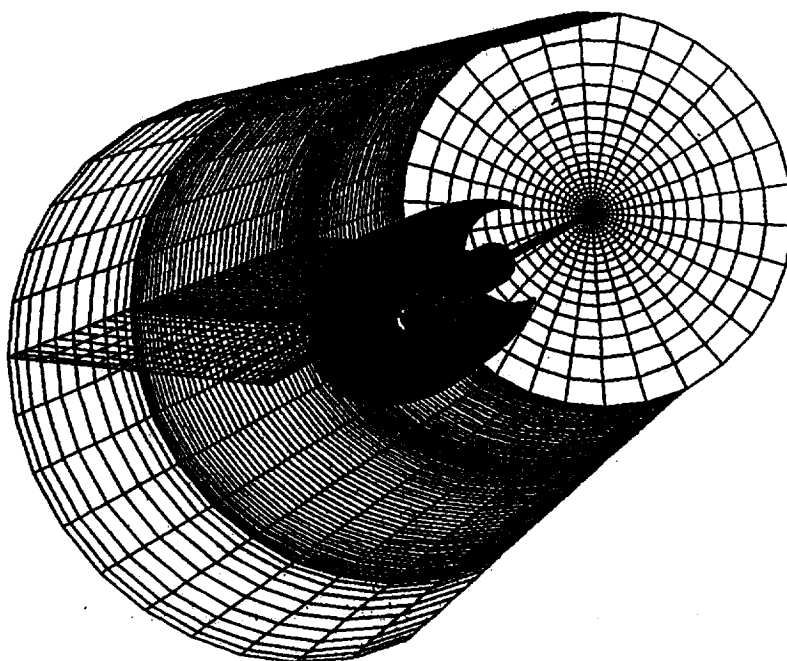
Figure 3.30 Multi–blocks volume grids for mock engine.

Another geometry modeled by this technique is the wind tunnel design. The boundary of this wind tunnel is given with a set of discrete data points (121 points). The curve is interpolated by the equations (3.6) ~ (3.8), and the data reduction algorithm is used to reduce the redundant control polygon. Figure 3.31 shows the interpolated BSpline curve with the reduced control polygon.

Figure 3.31 An interpolated BSpline curve for nozzle boundary.

Because this curve is parallel to z axis, one can utilize the "surface of revolution" algorithm (described in chapter two) to construct a NURBS surface by revolving the curve with respect to the z axis. The surface will be symmetric with respect to the z axis. One can apply the "scaling" algorithm to this NURBS surface with zero x, y scaling factor to generate another NURBS surface. This surface will be a degenerate surface which aligns to the z axis. This is demonstrated in Figure 3.32. After these two NURBS surfaces are obtained, one can perform the "ruled volume" algorithms (described in previous section) to construct the final NURBS 3D nozzle volume grid. Figure 3.33 shows the final nozzle.
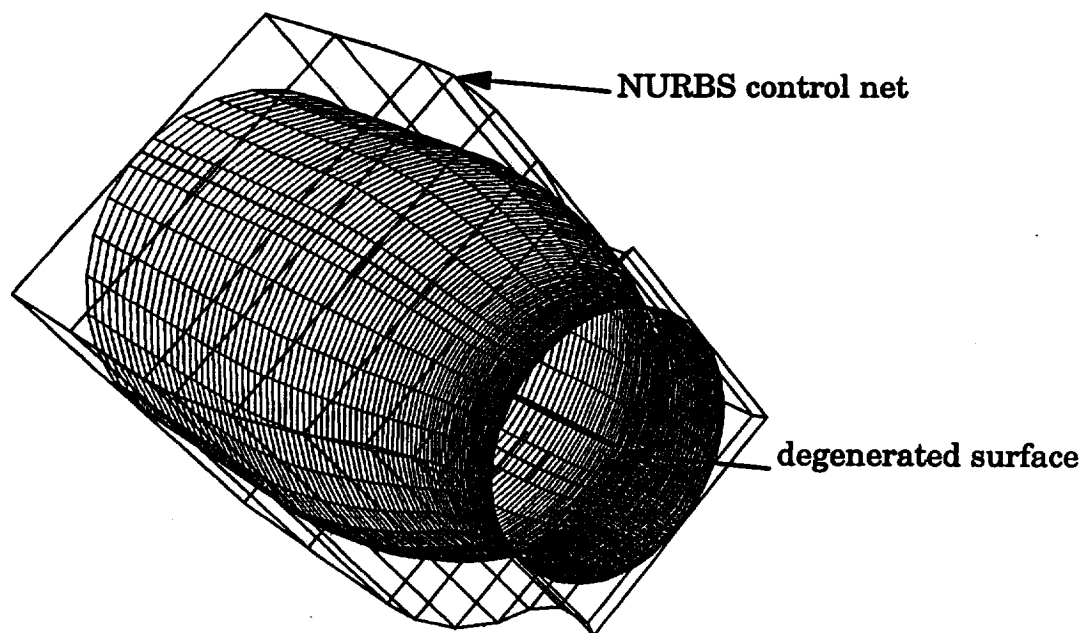
Figure 3.32 A nozzle geometry constructed by NURBS.
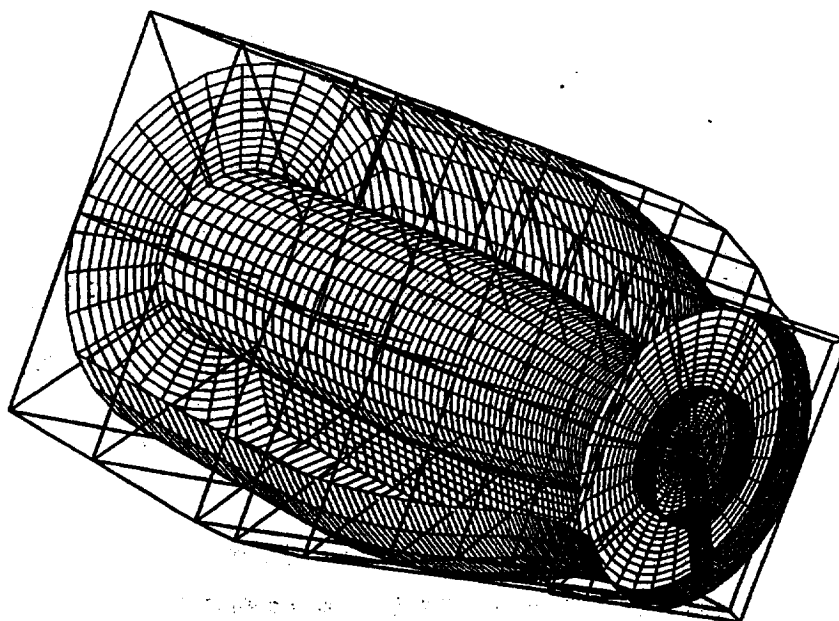
Figure 3.33 A 3D NURBS nozzle.

Projection and Inversion

After presenting the interpolation technique for modelling the free-form curve and surface, one may be interested to know how "well" the interpolated curve (or surface) describes the discrete data set. The algorithms to measure the deviation between the interpolated curve (or surface) and the sculptured data points are the "projection" and "inversion" algorithms. The definition of the projection problem can be stated as: Given an arbitrary point $P = (x, y, z)$ in 3D space, find a point on a NURBS curve or surface such that the distance between this point and point P can be the shortest. The definition of the inversion problem is that given the point P, assuming this point is on the NURBS entity (curve or surface), find the associated parametric values t (for curve) or s, t (for surface) such that one can utilize this parametric value to obtain the point P. More precisely, finding t such that $C(t) = P$ (for curve) or finding (s,t) such that $S(s,t) = P$ (for surface).

For the projection problem, if the point P is on the NURBS curve (or surface), then the shortest point would be the point P itself. Then, finding the parametric value for this P would become the inversion problem. Hence, the projection and inversion problems are the two sides of the same question. This question can then be combined with the two definitions and stated as: Given a point P (not necessarily lying on the NURBS entity), find the parametric value such that the parametric value can generate a point P^*, which is on the NURBS entity, and such that the distance of PP^* is also the shortest. If the point P is on the NURBS entity, then the P^* would be identical to point P (or the distance of PP^* would be zero). This is illustrated in Figure 3.34 for the curve case.
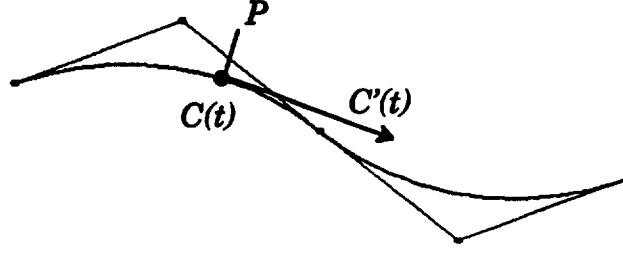
Figure 3.34 Illustration of projection and inversion for NURBS curve.

The algorithm is described as follows (refer to Figure 3.34 for the curve case): Let P be the point in 3D space, $C(t)$ be the point on the NURBS curve with parametric value t, and $C'(t)$ be the tangent vector associated with parametric value t. The value of t is obtained by solving the equation (5.1) formed by the inner product of the two vectors.

$$f(t) = C'(t) \cdot (C(t) - P) = 0 \qquad (3.14)$$

When the distance from point P to $C(u)$ is minimized, these two vectors will be orthogonal to each other, and hence, the inner production function $f(t)$ will be zero. The desired parametric value t which causes the shortest distance can be obtained by using the Newton Raphson iteration. The iterative equation is listed as equation (3.15).

$$t_{i+1} = t_i - \frac{f(t_i)}{f'(t_i)} = t_i - \frac{C'(t_i) \cdot (C(t_i) - P)}{C''(t_i) \cdot (C(t_i) - P) + |C'(t_i)|^2} \qquad (3.15)$$

The derivative of the NURBS curve, $C'(t_i)$ and $C''(t_i)$, can be found in [Ref 44,47]. The tolerance ε is set to check the convergence, and the stopping criteria can be set by either $| t_{i+1} - t_i | \leq \varepsilon$ or $| C(t_i) - P | \leq \varepsilon$. Upon convergence, t_i is the desired parametric value (for the inversion problem), and $C(t_i)$ is the projected point (for the projection problem).

The problems of projection and inversion for the NURBS surface are analogous. Because there are two independent variables (s, t) for a surface, it is necessary to create three vectors to form the equations. This yields the two inner production equations shown as (3.16) and (3.17).

$$f(s,t) = (S(s,t) - P) \cdot S_s(s,t) = 0 \qquad (3.16)$$
$$g(s,t) = (S(s,t) - P) \cdot S_t(s,t) = 0 \qquad (3.17)$$

The $S_s(s,t)$ is the derivative of a NURBS surface with respect to I direction evaluated at parametric value (s,t), while $S_t(s,t)$ is the derivative of a NURBS surface with respect to J direction evaluated at parametric value (s,t) [Ref 34,44]. When the distance between the point P to the point $S(s,t)$ is the shortest, equations (3.16) and (3.17) will be satisfied. The Newton iteration for solving equations (3.16) and (3.17) is shown in equation (3.18).

$$\begin{bmatrix} f_s(s_i,t_j) & f_t(s_i,t_j) \\ g_s(s_i,t_j) & g_t(s_i,t_j) \end{bmatrix} \begin{bmatrix} s_{i+1} - s_i \\ t_{j+1} - t_j \end{bmatrix} = \begin{bmatrix} -f(s_i,t_j) \\ -g(s_i,t_j) \end{bmatrix} \qquad (3.18)$$

Let vector R be $R=S(s,t)-P$. Then the elements of matrix on the left hand side in equation (3.18) can be rewritten as:

$$f_s(s_i \, , \, t_j) = |S_s|^2 + R(s_i \, , \, t_j) \cdot S_{ss} \qquad (3.19)$$
$$f_t(s_i \, , \, t_j) = S_s \cdot S_t + R(s_i \, , \, t_j) \cdot S_{st} \qquad (3.20)$$
$$g_s(s_i \, , \, t_j) = S_s \cdot S_t + R(s_i \, , \, t_j) \cdot S_{st} \qquad (3.21)$$
$$g_t(s_i \, , \, t_j) = |S_t|^2 + R(s_i \, , \, t_j) \cdot S_{tt} \qquad (3.22)$$

Also $\Delta t_j = t_{j+1} - t_j$, $\Delta s_i = s_{i+1} - s_i$. The convergence is achieved when the differences of Δt_j and Δs_i both are under the tolerance. The values of t_j and s_i are then the desired parametric values (for the inversion problem), and the point $S(s_i, t_j)$ is the projected point.

The algorithms of projection and inversion can be applied to many applications. For example, one can use the projection algorithm to measure the deviation between the discrete data set and the BSpline curve (or surface) obtained by the interpolation technique. This is done by projecting the discrete points to the interpolated curve. If the corresponding projected points are within the tolerance with respect to the original discrete points, then the interpolated BSpline curve is good enough to model the data set. The other example, as shown in Figure 3.35, projects an arbitrary curve on a NURBS surface. In this Figure, two points are selected on the NURBS surface. The inversion algorithm is then used to obtain the associated parametric values in the domain of the surface. A line joining these respective parametric values is evaluated to form the curve on this surface.
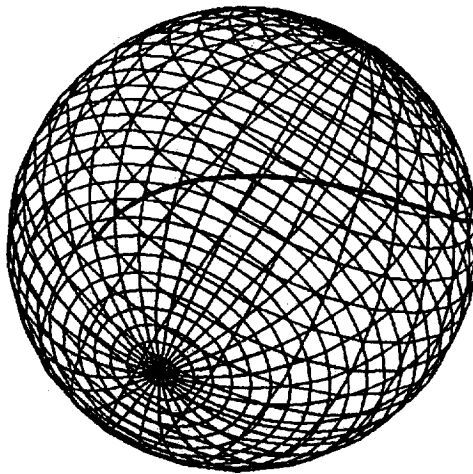
Figure 3.35 The projection of a curve to a NURBS surface.

CHAPTER IV

GRID REDISTRIBUTION AND EVALUATION

NURBS has been used in the CAD/CAM system for decades. It has been used for free–form surface representation and modeling for a long time. And from the generation and manipulation abilities described in previous chapters, one can realize that NURBS has become a CAD/CAM industry standard due to its versatile properties. However, there are several difficulties which inhibit people from fully utilizing this representation. The software tools available to designers for creating and reshaping such geometry are often inefficient for industrial parts, thereby consuming a large amount of designer time.

Obstacles of Using NURBS

The obstacles of using NURBS are discussed as follows:

Geometry Fidelity Maintenance

The NURBS representation is a parametric representation, and it is difficult to obtain the desired parametric value to maintain the critical geometric location. An example of this would be a NURBS representation which contains a sharp corner, which is defined as a slope discontinuity. How can the proper parametric value be obtained to maintain the original geometric information? Figure 4.1 shows the control polygon and its grid points on a NURBS composite curve. The curve does not maintain the sharp corner at the joint portion of the straight line and circular arc. Hence, the geometric fidelity is lost. The

challenge of this problem is to obtain the proper parametric value *t* such that the final curve would be the one shown in Figure 4.2.
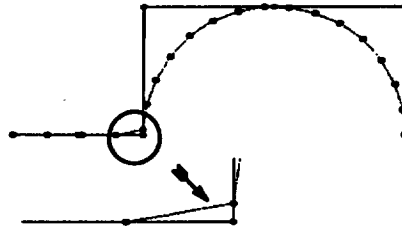
Figure 4.1 Improper parametric values lose the geometry fidelity.
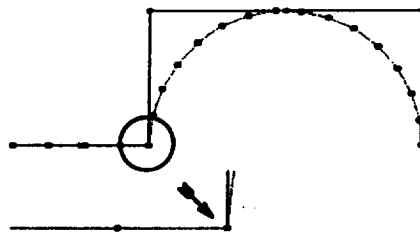
Figure 4.2 Proper parametric values keep the geometry fidelity.

Distribution Control of the Grid Points on a 3D Physical Space

The NURBS entity (curve, surface, or volume) is presented as a parametric format, and the grid point on a NURBS entity is generated by evaluating the parametric value *t* (or *s*, *t* for surface, *s*, *t*, *u* for the volume). However, the designer desires the distribution of the grid points on the physical NURBS entity, not the distribution of the parametric values. For example, evenly distributed parametric values *t* may not result in a sequence of evenly distributed grid points of *C(t)* on the physical NURBS curve. Finding the correct parametric values to obtain the desired distribution on 3D physical space has presented a problem to engineers for a long time. Figure 4.3 shows the control polygon and grid points on a NURBS curve. Even though the parametric val-

ues used for evaluation of this curve are distributed evenly in parametric domain, one can observe that the grid points are actually packed toward the middle. The location of the control polygon, the *weights* and even the knot vector are all possible factors in controlling the final NURBS entity. The challenge of this problem is, without altering the NURBS definition (control polygon, *weights* and knot vector), calculating the proper parametric values to obtain the desired distribution on the physical NURBS entity as shown in Figure 4.4.
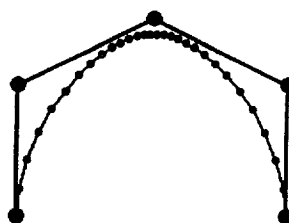
Figure 4.3 Illustration of undesirable distribution on NURBS curve.
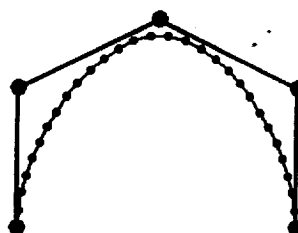
Figure 4.4 Illustration of desirable distribution on NURBS curve.

Bad Parameterization

It has been previously mentioned that, setting the communication between CAD/CAM and grid generation system is a ideal for shortening the entire CFS process, because most the geometry of interest is created from the CAD/CAM package. However, when importing the geometry from a CAD/CAM package, the engineer has to handle this geometry with all information

pre-defined. Unfortunately, that predefined information may lead to unexpected surface grids. Taking the Figure 4.5 as an example, this surface is obtained from the package *CATIA* (a product of the *IBM* CAD system). The surface is presented as a NURBS surface. From the Figure 4.5, one can see that the control polygons are clustered in the middle of the surface in the J direction as well as the lower index of the I direction of the surface. The knot vectors are set evenly in both I and J directions. These NURBS definitions (both the location of control net and the setting of the knot vectors) make the surface grid lines packing towards the clustered control nets. This unexpected and undesired packing of the surface grid lines will produce a poor quality grid which might result in a divergence of CFS simulation. The situation is referred to as "bad parameterization". This is the first penalty one has to pay for the geometry communication between systems due to an unexperienced CAD/CAM designer. Also, the improper geometry definitions happen frequently during this communication. Since the engineers from grid generation did not participate in designing the geometry when it was in CAD system, the challenge becomes finding a solution to obtaining the desired distribution without altering the geometry shape as shown in Figure 4.6.

Another penalty is related to geometry manipulation. One of the NURBS properties is the local control property which states that altering the location of the control points or the values of *weight* will only modify the shape of geometry locally. This local control property is very attractive to the designer because the modification will not change the entire geometry. However, improper manipulation of the NURBS control points or *weights* may lead to poor grid points (lines) packing, and this situation also results in "bad parameterization". Maintaining the desired shape of geometry (after changing the con-

trol points or *weights*) and the proper geometric parameterization is challeng-
ing. Figure 4.7 shows the control net and the surface grid lines of a NURBS
surface after increasing the *weight* of one of the control points. One can see
that, the surface grid lines are packed toward the control points with in-
creased *weight*. The challenge of this problem is finding a better parameter-
ization without altering the desired shape of the geometry as shown in Figure
4.8.
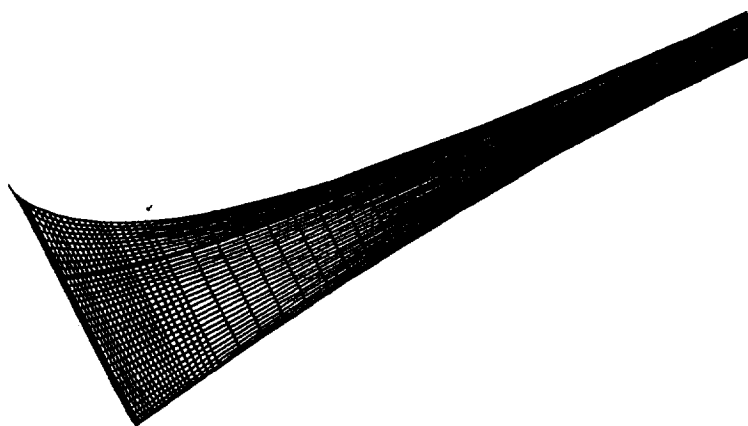
Figure 4.5 Bad locations of a NURBS control net.
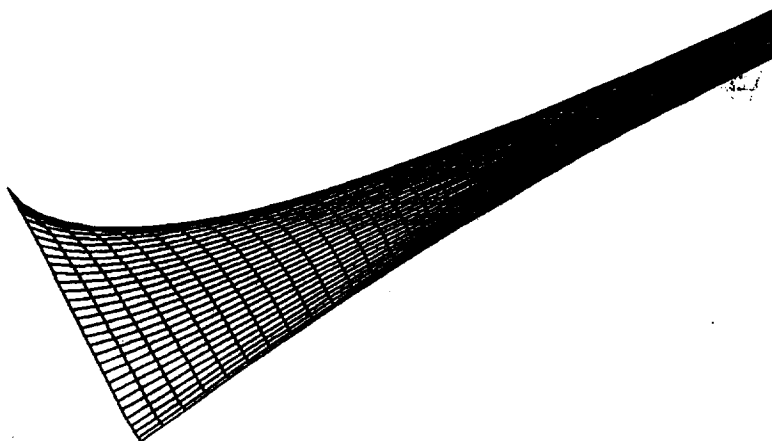
Figure 4.6 A uniform distribution NURBS surface.
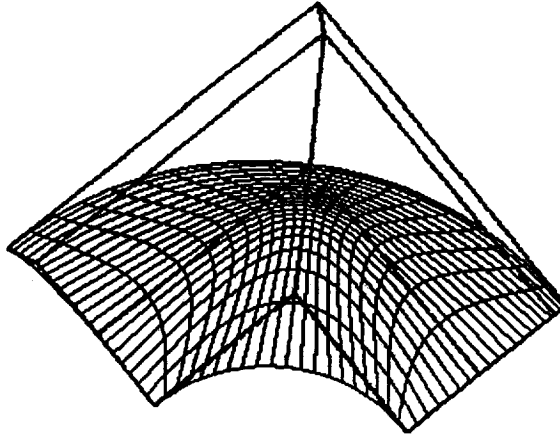
Figure 4.7 Improper manipulation leading to bad parameterization.
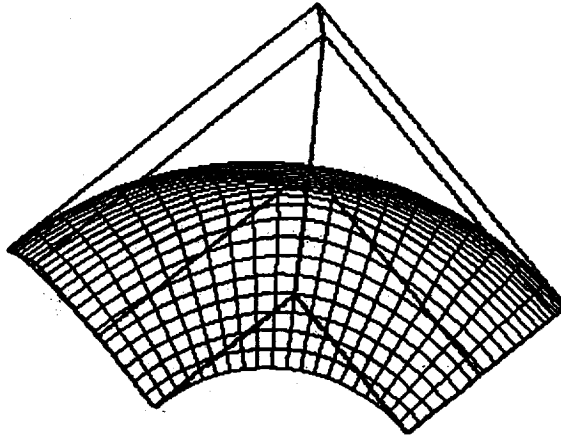
Figure 4.8 Improved parameterization surface after re–parameterization.

Computation Intensive for NURBS Evaluation

The other complaint, which is not related to geometry communication, for using the NURBS modeling is the computationally intense evaluation. Given the control polygon d and *weights* W from equations (2.1)~(2.4), the challenge for efficient evaluation is finding the basis function $N_i^k(t)$ at particular value t as quickly as possible, since this basis function is defined recursively. Of the many approaches to evaluating the grid points on a NURBS entity,

the "de Boor" [Ref 18] method is the most commonly used. Even though many computer languages, such as C and *Pascal*, allow the recursive call, evaluating this basis function with an internal recursive call needs substantial memory and computation time. Also, when the *order(s)* or the dimension of the NURBS control points is high, especially for the case of a 3D NURBS control volume, the evaluation will take significant computation time. Avoiding the recursive call and quickly evaluating the grid points on a NURBS entity is an important issue.

Strategies for Overcoming NURBS Difficulties

The strategies for overcoming the problems associated with using NURBS are presented in this section.

As one can understand from equations (2.1) ~ (2.4), the NURBS is defined in a parametric format. Thus, any parametric value will create a physical grid point on a NURBS entity. Taking the NURBS curve as an example, any value t in parametric space, which must be inside the domain of the knot vector, will result in a point $C(t)$ in the physical space. In other words, the evaluation domain for any parametric value t must be located in the range of the knot vector of T, $t \in [T_{k-1}, \dots, T_{n+1}]$. The first problem — maintaining the NURBS geometric fidelity, is related to the knot vector.

Maintaining the Sharp Corner with NURBS

In order to know how to maintain the fidelity of the geometry, it is necessary to know the relationship between the NURBS entity and the knot vectors. This is illustrated as follows: a NURBS curve has *order* equal to 3 and control points a, b, c, d and e, as shown in Figure 4.9. Its associated knot vector is [0., 0., 0., 0.3333, 0.6667, 1., 1., 1.]. The points *m1* and *m2* are the points

evaluated with the parametric values t equal to *0.3333* and *0.6667* (notice that these two values are exactly the same as the interior knot values of the knot vector). The arc *am1* is created from those parametric values located in the first knot span [0., 0.3333], and the arc *m1m2* is created from the parametric values located in the second knot span [0.3333, 0.6667], while the last arc *m2e* is created from the parametric values located in the last knot span [0.6667, 1.0]. In each knot span, the associated NURBS arc is a smooth curve. If the knot value $t_1 = 0.3333$ increases, and the knot value $t_2 = 0.6667$ decreases (but $t_1 \neq t_2$), then the arc *m1m2* is "shrunk". Before the t_1 and t_2 values collide, the segments are all C^2 continuously differentiable due to the *order* equal to 3. In other words, in each arc segment, there is no discontinuous point. However, as one can imagine, when the t_1 and t_2 value collide, the arc *m1m2* becomes a singularity point, and the discontinuous problem may arise. It is well known [Ref 23,34,44] that the continuity of a NURBS entity at knot value t is equal to $(k-1-m)$ where k is the *order* and m is the multiplicity of the knot value. Hence, if t_1 and t_2 collide, then the multiplicity of this knot value becomes 2, and the continuity is $C^{(3-1-2)} = C^0$, which defines the sharp corner (discontinuous point). This situation occurs when the two NURBS entities are joined together by the "composite" algorithm described in chapter 2. In that algorithm, the multiplicity of the knot value has been set to (*order*–1) in order to maintain the discontinuous portion.

Knowing the reason for the creation of the sharp corner, the strategy for maintaining the discontinuous point is as follows: when the sequence of parametric values t_i are generated, one should first check the domain of the knot vector (for this example, the domain of knot vector [0.0, 0.3333, 0.6667, 1.0]). If any knot value T_i has multiplicity m which gives $(k-1-m)$ equal to *0*, then

one should chose the parametric value t_j with the absolute minimum difference of $(t_j - T_i)$ and assign T_i to t_j. For example, if the knot vector for the curve shown in Figure 4.1 is set [0., 0., 0., 0.25, 0.52, 0.52, 0.75, 0.75, 1., 1., 1.0], and the user would like to distribute 11 points on the curve with even spacing in parametric space (t_i = 0.1i i=0, ..., 10), since T_5 = T_6 =0.52 (with multiplicity equal to 2), then t_5 = 0.5 must be adjusted to the value of T_5 = T_6 =0.52. The same procedure should apply to the parametric value t_7 = 0.7 (or t_8 = 0.8, since $|t_7 - T_7| = |t_8 - T_7|$), updating it to the value of T_7 = T_8 =0.75. After this process, the parametric sequence is ensured to maintain all possible discontinuous points.
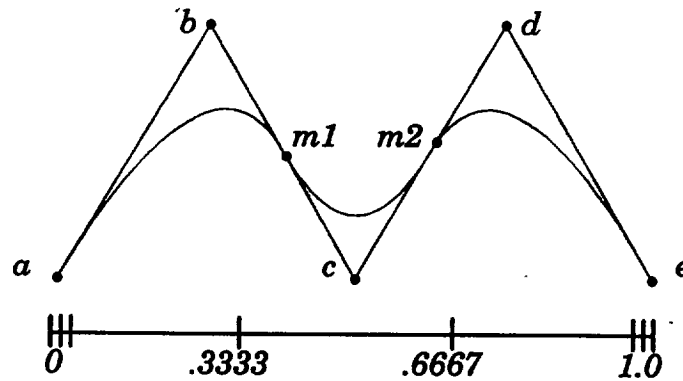
Figure 4.9 The relationship between knot value and the NURBS curve.

It is necessary to add a "physical" checking algorithm to find out the "real" sharp corners after the previous procedures. In many cases, the knot value which has multiplicity greater than 1 does not intend to represent the discontinuous point. For example, the circular arc shown in Figure 2.2 has the knot vector set [0., 0., 0., 0.5, 0.5, 1., 1., 1.]. Even though the knot value 0.5 has the multiplicity 2, according to the previous discussion, it should have a sharp corner because $C^{(3-1-2)} = C^0$. However, one knows there are no discontinuous points in a semi-circle. There is no harm in setting any parametric

values to the knot value 0.5, but to avoid taking the point M as the sharp corner, one should add a distinguishing algorithm described as follows: If t_i is the suspect parametric value which might create the discontinuous point, then setting three additional parametric values t_{i-1}, t_{i+1} and t_{i+2}, with t_{i-1} slightly smaller than t_i, and t_{i+1}, t_{i+2} slightly greater than t_i ("slightly" can be defined as the distance of t_i and t_{i-1} to be 1.0e–7) will discourage discontinuity. Evaluating with these four parametric values will yield four points — $C(t_{i-1})$, $C(t_i)$, $C(t_{i+1})$ and $C(t_{i+2})$ in the physical curve entity. These four points will form two angles, θ_1 and θ_2, as shown in Figure 4.10. If the absolute difference of these two angles is greater than the tolerance (for example, 5^0), then this parametric t_i is indeed a value which can create a real physical sharp corner.
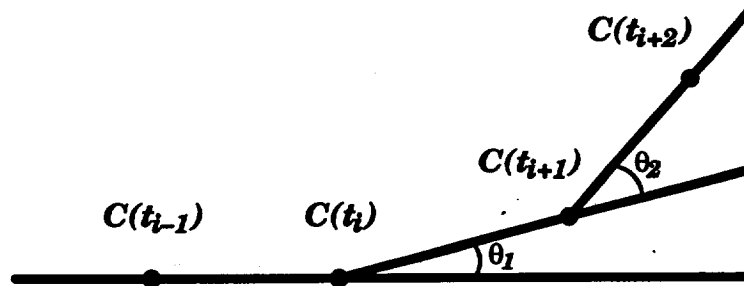
Figure 4.10 Definition of a discontinuous point.

Re–Parameterization Algorithm

The second and third barriers can be overcome by the re–parameterization algorithm. This algorithm allows the user to determine the parametric values which yield the desired distribution on physical NURBS entities. Two approaches are introduced: one is performed with an iterative method, and the other is implemented by linear interpolation.

Before discussing the algorithms, it is necessary to define several notations. For a NURBS curve with resolution ni, there are several $1D$ arrays which need to be defined:

(1) $cs_1(i)$, $i=1, ..., ni$ be the parametric values associated with the desired distribution of the curve in physical space;

(2) $cs_2(i)$, $i=1, ..., ni$ be the normalized chord length of the curve with desired distribution (for example, $cs_2(i) = (i-1)/ (ni-1)$ for even distribution);

(3) $cs_3(i)$, $i=1, ..., ni$ be the normalized chord length of the curve evaluated at parametric values $cs_1(i)$, $i=1, ..., ni$.

The explanation of these $1D$ arrays are referred to Figure 4.3 and 4.4. Consider the example shown in Figure 4.3. If the designer would like to have the final curve as shown in Figure 4.4, then $cs_2(i)$ will be a $1D$ array which contains the distribution packing evenly, and $cs_1(i)$ are the parametric values which are to be determined, such that the $cs_3(i)$, the normalized chord length of final curve, would be the same as $cs_2(i)$ (or $|cs_2(i) - cs_3(i)|$ be minimized for all $i=1,.., ni$).

For the NURBS tensor product surface with resolution ni by nj, the $2D$ arrays are defined as follows:

(1) $(ss_1(i,j), st_1(i,j))$, $i=1, ..., ni, j=1, ..., nj$ be the parametric distribution mesh associated with the desired distribution of the surface in physical space;

(2) $(ss_2(i,j), st_2(i,j))$, $i=1, ..., ni, j=1, ..., nj$ be the normalized chord length of the surface with desired distribution in direction I and J respectively;

(3) $(ss_3(i,j), st_3(i,j))$, $i=1, ..., ni, j=1, ..., nj$ be the normalized chord length of the surface evaluated at parametric values $(ss_1(i,j), st_1(i,j))$ $i=1, ..., ni, j=1, ..., nj$.

Consider Figure 4.7 and 4.8 as examples. If the designer would like to have the final surface, as shown in Figure 4.7, then $(ss_2(i,j), st_2(i,j))$ would be a

2D array which contains the even distribution, and $(ss_1(i,j), st_1(i,j))$ would be the parametric values which are to be determined such that $(ss_3(i,j), st_3(i,j))$, the normalized chord length of final surface, would be the same as $(ss_2(i,j), st_2(i,j))$ or within certain tolerance.

For the NURBS tensor product volume with resolution ni, nj and nl, the *3D* arrays are defined as follows:

(1) $(vs_1(i,j,l), vt_1(i,j,l), vu_1(i,j,l))$, $i=1$, ..., ni, $j=1$, ..., nj and $l=1$, ..., nl be the parametric distribution volume associated with the desired distribution of the volume in physical space;

(2) $(vs_2(i,j,l), vt_2(i,j,l), vu_2(i,j,l))$, $i=1$, ..., ni, $j=1$, ..., nj and $l=1$, ..., nl be the normalized chord length of the volume with desired distribution in direction I, J and L respectively;

(3) $(vs_3(i,j,l), vt_3(i,j,l), vu_3(i,j,l))$, $i=1$, ..., ni, $j=1$, ..., nj and $l=1$, ..., nl be the normalized chord length of the volume evaluated at parametric values $(vs_1(i,j,l), vt_1(i,j,l), vu_1(i,j,l))$ $i=1$, ..., ni, $j=1$, ..., nj and $l=1$, ..., nl.

The illustration of these 3D arrays are analogous to curve and surface cases.

1> Iteration for Re-Parameterization Algorithm.

The procedure for the reparameterization algorithm for a NURBS curve is stated as follows: The first step is to initialize the values of $cs_1(i)$ by setting $cs_1(i) = cs_2(i)$. Once $cs_2(i)$ are determined, the second step is to evaluate the NURBS curve using the parametric values of $cs_1(i)$ and calculate the normalized chord length, cs_3, of this curve. Generally, $cs_3(i)$ will not be the same as $cs_2(i)$. If any value of $cs_3(i)$, say $cs_3(I)$, is greater (or smaller) than $cs_2(I)$ by the tolerance, because $cs_3(I)$ is obtained by evaluating the NURBS curve with the parametric value of $cs_1(I)$, then the value of $cs_1(I)$ should be decreased (or in-

creased if it is smaller). This process needs to be checked with all the values of $cs_2(I)$ and $cs_3(I)$ iteratively until all the differences of $|cs_2(i) - cs_3(i)|$ are all within the tolerance for all $i=1,...,ni$. More precisely, the iterative algorithm can be described in the computer pseudocode shown as *Algorithm 4.1*.

Algorithm 4.1

```
for (j=0; j<max_iteration; j++) {
    check whether |cs₂(i)–cs₃(i)| < tolerance, 0<i<ni
    if (yes)   break the for loop and program stop;
    else
    {
            for (i=1;i<ni;i++) {
                if (cs₂(i) < cs₃(i))
                    cs₁(i) += (cs₂(i) – cs₃(i))*(cs₁(i)–cs₁(i–1));
                else
                    cs₁(i) += (cs₂(i) – cs₃(i))*(cs₁(i+1)–cs₁(i));
            }
            Repeat Step 2.
    }
}
```

The *max_iteration* and *tolerance* are set according to different request. If the *tolerance* is set too small, the program needs more iterations. For the examples shown in Figure 4.3 and 4.4, the *tolerance* is set as $1.0e{-}6$, and it needs 51 iterations.

The reparameterization algorithm for the NURBS surface is analogous to the curve algorithm. The first step is to initialize the values of $(ss_1(i,j), st_1(i,j))$ as the desired distribution mesh $(ss_2(i,j), st_2(i,j))$. The second step is to evaluate the surface with these parametric values to obtain $(ss_3(i,j), st_3(i,j))$. Then the normalized chord length of the entire surface is calculated for $(ss_3(i,j), st_3(i,j))$. Again, $(ss_3(i,j), st_3(i,j))$ will be different than the desired distribution mesh of $(ss_2(i,j), st_2(i,j))$. The idea of adjusting the parametric mesh

is simply the extension of 1D arrays of the curve to 2D arrays and is presented in the computer pseudocode shown as *Algorithm 4.2*.

Algorithm 4.2

```
for (k=0; k<max_iteration; k++) {
    check whether |ss2(i,j)-ss3(i,j)| < tolerance for 0<i<ni, 0<j<nj
    if (no) {
        for (j=0;j<nj;j++)
        for (i=1;i<ni-1;i++) {
            if (ss2(i,j) < ss3(i,j))
                ss1(i,j) += (ss2(i,j)-ss3(i,j))*(ss1(i,j)-ss1(i-1,j));
            else
                ss1(i,j) += (ss2(i,j)-ss3(i,j))*(ss1(i+1,j)-ss1(i,j));
        }
    }
    check whether |st2(i,j)-st3(i,j)| < tolerance for 0<i<ni, 0<j<nj
    if (no) {
        for (i=0;i<ni;i++)
        for (j=1;j<nj-1;j++) {
            if (st2(i,j) < st3(i,j))
                st1(i,j) += (st2(i,j)-st3(i,j))*(st1(i,j)-st1(i,j-1));
            else
                st1(i,j) += (st2(i,j)-st3(i,j))*(st1(i,j+1)-st1(i,j));
        }
    }
    if (both checking return yes)
            break the loop and program stop.
    else    Repeat Step 2.
}
```

From this pseudocode, one can realize that adjusting the parametric value in the I direction for $ss_1(i,j)$ will effect the calculating of $st_1(i,j)$, and adjusting the value for $st_1(i,j)$ will effect the calculating of $ss_1(i,j)$ too. This situation is easy to understand because the surface is defined as a tensor product surface. However, due to this mutual effect, the convergence of the entire procedure needs more computation time that that of a curve. For the examples

shown in Figures 4.7 and 4.8, the *tolerance* is set as 1.0e–5, and it needs 216 iterations for the 31 by 31 surface resolution. If the tolerance is set to 1.0e–6, the number of iterations increases to 1500 iterations for the same resolution.

The volume reparameterization algorithm for NURBS can be extended from *2D* surface arrays to *3D* volume arrays. However, since the NURBS volume is also defined as a tensor product format, adjusting the parametric values of any one of the directions (I, J or K) will automatically effect the calculation of the other two directions. This situation makes it difficult for the algorithm to reach the convergence of a small tolerance, and it also greatly increases the computation time for one iteration of volume evaluation. Due to this drawback, the iterative algorithm is seldom utilized for the NURBS volume case. An alternative way for this algorithm is the linear interpolation approach described in the next section.

2> Linear Interpolation for Re–Parameterization Algorithm.

For the curve case, in order to obtain $cs_3(i)$, the initial values of $cs_1(i)$ must be initialized. One can set $cs_1(i)$ as evenly distributed initially, evaluate the curve with these parametric values to obtain the final curve, then normalize the entire chord length to obtain $cs_3(i)$. Generally, the $cs_3(i)$ will be different than $cs_2(i)$. How does one adjust $cs_1(i)$ to have new $cs_3(i)$ which is close to $cs_2(i)$? The entire procedure is described in the *Algorithm 4.3*.

Algorithm 4.3

```
for (i=1; i<(ni–1); i++) {
    find the location of j such that cs₃(j) ≤ cs₂(i) < cs₃(j+1).
    let α = cs₂(i) – cs₃(j);  β = cs₃(j+1) – cs₂(i);
}    cs(i) = (cs₁(j+1)*α + cs₁(j)*β) / (α + β);
```

The theory behind this pseudocode is the simple "linear–interpolation". The final $cs(i)$ should replace $cs_1(i)$ as the final desired parametric distribution. Using this distribution to evaluate the NURBS curve shown in Figure 4.3 will yield the result shown in Figure 4.4 — which has the even distribution (point packing) on the physical curve. The linear interpolation may not be very accurate, especially when the resolution is very low. Therefore, if the final curve does not achieve the desired distribution, one should repeat this procedure again. According to research experience, the result of this algorithm is very satisfactory after 3 iterations.

For the surface case, the procedure is similar to the NURBS curve. One can set $(ss_1(i,j),\ st_1(i,j))$ as evenly distributed initially, then evaluate the surface with these parametric values to obtain $(ss_3(i,j),\ st_3(i,j))$. Most likely $(ss_3(i,j),\ st_3(i,j))$ will be different than $(ss_2(i,j),\ st_2(i,j))$. The procedure to adjust $(ss_1(i,j),\ st_1(i,j))$ so that $(ss_3(i,j),\ st_3(i,j))$ can be as close as to $(ss_2(i,j),\ st_2(i,j))$ is presented in *Algorithm 4.4* as a computer pseudocode:

Algorithm 4.4

```
for (j=1; j<(nj–1); j++)
for (i=1; i<(ni–1); i++) {
    search the index of I, J such that (ss₂(i,j), st₂(i,j)) is located within the
        cell of (ss₃(I, J),st₃(I,J)), (ss₃(I+1,J),st₃(I+1,J)), (ss₃(I, J+1),st₃(I, J+1))
        and (ss₃(I+1,J+1),st₃(I+1, J+1))
    let (ss₂(i,j),st₂(i,j)) = (1–α)(1–β)(ss₃(I, J),st₃(I, J)) + (1–α)β(ss₃(I, J+1),
        st₃(I, J+1))+α(1–β)(ss₃(I+1, J), st₃(I+1, J))+αβ(ss₃(I+1, J+1), st₃(I+1,
        J+1)) and solve for α and β ;
    new (ss(i,j),st(i,j)) = (1–α)(1–β)(ss₁(I, J), st₁(I, J)) + (1–α)β(ss₁(I, J+1),
        st₁(I,J+1)) + α(1–β)(ss₁(I+1, J), st₁(I+1, J)) + αβ(ss₁(I+1, J+1), st₁(I+1,
        J+1));
}
```

One can verify that the bi–linear interpolation is used in this algorithm for finding the new parametric values. Even this bi–linear interpolation is more accurate than linear interpolation, but when the resolutions are not high

enough, two or three iterations may be needed to improve the quality. After this algorithm, the final $((ss(i,j),\ st(i,j))$ should update $((ss_1(i,j),\ st_1(i,j))$ and then $((ss_1(i,j),\ st_1(i,j))$ is the desired distribution mesh in which to generate the desired surface grids shown in Figure 4.8. In this algorithm, the Newton method [Ref 19] is used for solving the non–linear equation for multi–variables α and β.

After knowing how the algorithms work for curve and surface, the algorithm for the volume is analogous. The only thing different is that the trilinear interpolation should be used for solving the equation (4.1) by Newton iteration for variables α, β and γ.

$$
\begin{aligned}
(vs_2(i,j,l),vt_2(i,j,l),vu_2(i,j,l)) &= (1-\alpha)(1-\beta)(1-\gamma)(vs_3(I,J,L),vt_3(I,J,L),vu_3(I,J,L)) \\
&+ \alpha(1-\beta)(1-\gamma)(vs_3(I+1,J,L),vt_3(I+1,J,L),vu_3(I+1,J,L)) \\
&+ (1-\alpha)\beta(1-\gamma)(vs_3(I,J+1,L),vt_3(I,J+1,L),vu_3(I,J+1,L)) \\
&+ (1-\alpha)(1-\beta)\gamma(vs_3(I,J,L+1),vt_3(I,J,L+1),vu_3(I,J,L+1)) \\
&+ \alpha\beta(1-\gamma)(vs_3(I+1,J+1,L),vt_3(I+1,J+1,L),vu_3(I+1,J+1,L)) \\
&+ \alpha(1-\beta)\gamma(vs_3(I+1,J,L+1),vt_3(I+1,J,L+1),vu_3(I+1,J,L+1)) \\
&+ (1-\alpha)\beta\gamma(vs_3(I,J+1,L+1),vt_3(I,J+1,L+1),vu_3(I,J+1,L+1)) \\
&+ \alpha\beta\gamma(vs_3(I+1,J+1,L+1),vt_3(I+1,J+1,L+1),vu_3(I+1,J+1,L+1))
\end{aligned}
\tag{4.1}
$$

After α, β and γ are obtained, the new parametric values are determined as shown in equation (4.2).

$$
\begin{aligned}
(vs(i,j,l),vt(i,j,l),vu(i,j,l)) &= (1-\alpha)(1-\beta)(1-\gamma)(vs_1(I,J,L),vt_1(I,J,L),vu_1(I,J,L)) \\
&+ \alpha(1-\beta)(1-\gamma)(vs_1(I+1,J,L),vt_1(I+1,J,L),vu_1(I+1,J,L)) \\
&+ (1-\alpha)\beta(1-\gamma)(vs_1(I,J+1,L),vt_1(I,J+1,L),vu_1(I,J+1,L)) \\
&+ (1-\alpha)(1-\beta)\gamma(vs_1(I,J,L+1),vt_1(I,J,L+1),vu_1(I,J,L+1)) \\
&+ \alpha\beta(1-\gamma)(vs_1(I+1,J+1,L),vt_1(I+1,J+1,L),vu_1(I+1,J+1,L)) \\
&+ \alpha(1-\beta)\gamma(vs_1(I+1,J,L+1),vt_1(I+1,J,L+1),vu_1(I+1,J,L+1)) \\
&+ (1-\alpha)\beta\gamma(vs_1(I,J+1,L+1),vt_1(I,J+1,L+1),vu_1(I,J+1,L+1)) \\
&+ \alpha\beta\gamma(vs_1(I+1,J+1,L+1),vt_1(I+1,J+1,L+1),vu_1(I+1,J+1,L+1))
\end{aligned}
\tag{4.2}
$$

This approach, compared to the volume case of the iterative algorithm, is much less computationally expensive.

Figure 4.11 and 4.12 show the results for a volume grid re–parameterization algorithm.
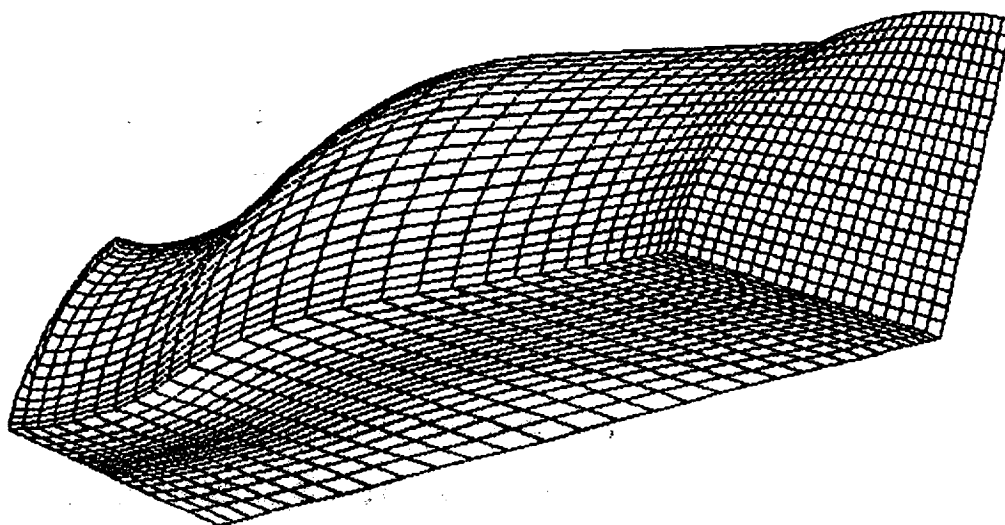
Figure 4.11 A volume grid with undesirable packing.
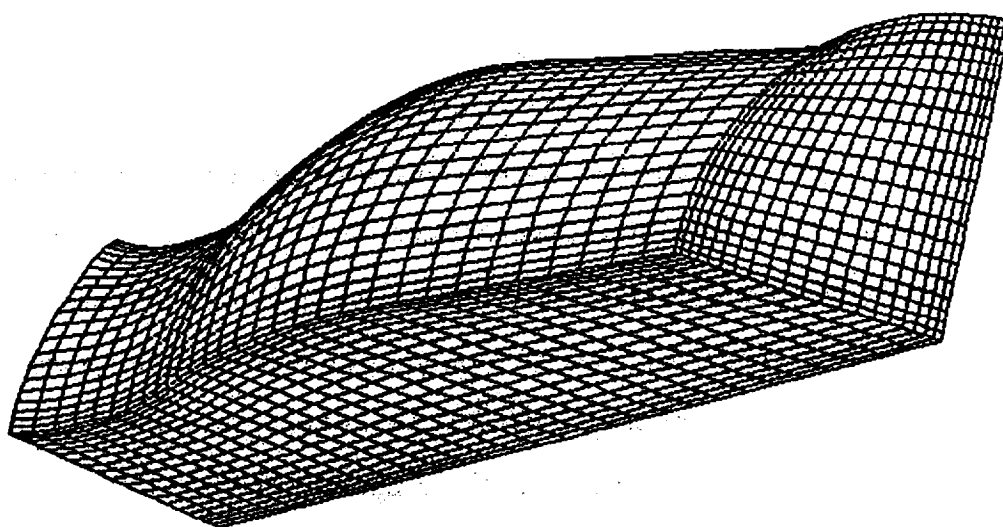
Figure 4.12 A volume grid with desirable packing after re–parameterization.

Singularity Control

As one examines these re–parameterization algorithms, one can easily find out that there is a serious flaw which may lead these algorithms to fail. This flaw is related to the "singularity" problem. Taking a NURBS curve as an example, if all the points on this curve collapse to one point, then this NURBS curve is a singular curve. The same definition can be applied to NURBS surface and volume. If any of the iso–parametric line on a NURBS surface collapses to one point, then this line is defined as singular line. For a volume, if any of the iso–plane on a NURBS volume collapses to a line, then this plane is defined as singular plane. When these singularity problems occur, the total arc length from calculating it according to the re–parameterization algorithms will be zero. Since the normalized arc lengths are obtained by dividing the total arc length to the individual ones, this will lead to the operation of 0/0 which is mathematically undefined and can not be implemented by any computer language. These singularity problems happen in many cases. For example, the surface grid which represents the canopy of a aircraft has a singular line at the nose position; a surface grid which represents the missile has a singular line in the nose position; and the volume grid of a cylinder (or any cylindrical pipe) has a singular plan at the axial direction. From these examples, one notices that the singularity problems occured because the control points collapse to one point (for the surface case) or one line (for the volume case). While evaluating the NURBS entities with certain parametric values by utilizing these collapsed control points, the singularity problem arises. Hence, it is necessary to enhance the algorithms to handle this problems.

The strategy of the enhancement algorithm is related to the machine accuracy (or called machine precision). The machine accuracy, commonly rep-

resented as symbol ε, is defined as the smallest positive real number such that $1 + \varepsilon > \varepsilon$. On the *Silicon Graphics Personal Iris*, this number is equal to 1.0^{-16} (double precision). In many numerical simulations, this number is needed to represent the finite precision arithmetic of the computer architecture. For example, the stopping criteria of an iteration scheme is dependent upon the machine accuracy. A variable expected to be *zero* in numerical representation may not be reached due to the finite precision of the computer memory representation. Hence, in many numerical applications, the exact *zero* is replaced by a value related to ε, say if a variable is less than $\sqrt{\varepsilon}$, then this variable is assumed to be equal to *zero*. This concept is also utilized to avoid the singularity problems. Taking the example of the NURBS surface with a singular line shown in Figure 4.13, the grid line evaluated with the parametric values of $(ss_1(0,j), st_1(0,j) j=0, ..., nj$ will shrink to one singular point due to the control vertices d_{0j} collapsing to one point. However, if one perturbes these parametric values by a small values, say $\sqrt{\varepsilon}$, and then re–evaluates the surface, the returned grid line will not be the same as the singular one since these parametric values are no longer exact *zero*. Instead, it will return a grid line with a small but recognizable total arc length. Even though the total arc length is small, the normalization process will make the values of $(ss_3(0,j), st_3(0,j)$ $j=0,...,nj$ to 0.0 ~ 1.0 and will avoid the uncertain situation of 0/0. This procedure is shown as the *Algorithm 4.5*.

Algorithm 4.5

```
for (j=0; j<nj; j++) {
    ss1(0,j) += ε; ss1(ni–1,j) –= ε;
}
for (i=0; i<ni; i++) {
    st1(i,0) += ε; st1(i,nj–1) –= ε;
}
```

Figure 4.14 illustrates the result of this algorithm applied to a NURBS surface with a singular line.
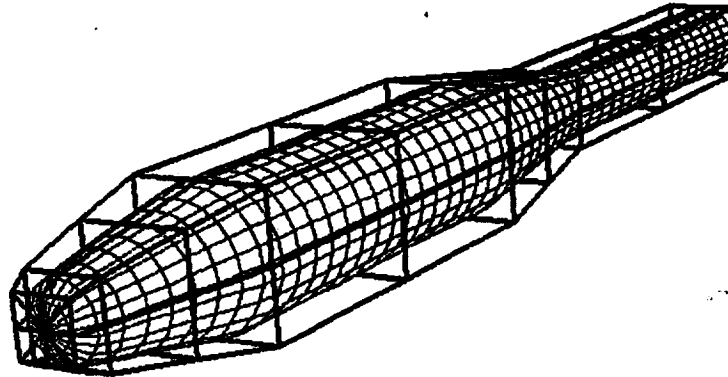
Figure 4.13 A NURBS surface with a singular line.
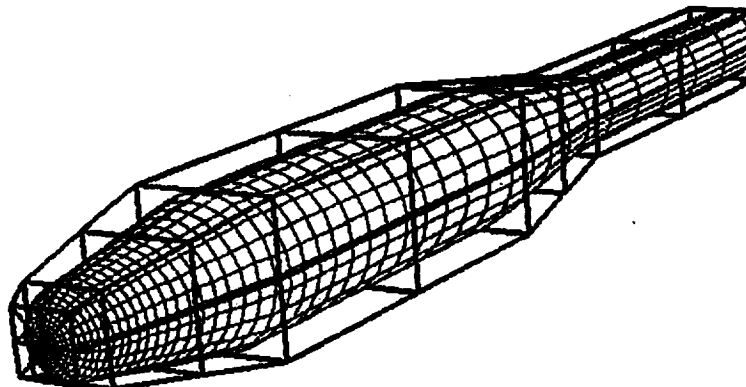
Figure 4.14 Reparameterization for a NURBS surface with singularity.

The associated approach can be applied to 3D NURBS volume. These "offset" procedures for the 3D parametric values can be summarized in *Algorithm 4.6*. Figure 4.15 shows a 3D NURBS cylindric pipe evaluated with even parametric values. Notice that in its L direction, the surface degenerates to a singer line. The result of reparameterization for this volume is shown in Figure 4.16.

Algorithm 4.6

```
for (k=0; k<nk; k++)
for (j=0; j<nj; j++) {
    vs₁(0,j,k) += ε; vs₁(ni–1,j,k) –= ε;
}
for (k=0; k<nk; k++)
for (i=0; i<ni; i++) {
    vt₁(i,0,k) += ε; ct₁(i,nj–1,k) –= ε;
}
for (j=0; j<nj; j++)
for (i=0; i<ni; i++) {
    vu₁(i,j,0) += ε; vu₁(i,j,nj–1) –= ε;
}
```
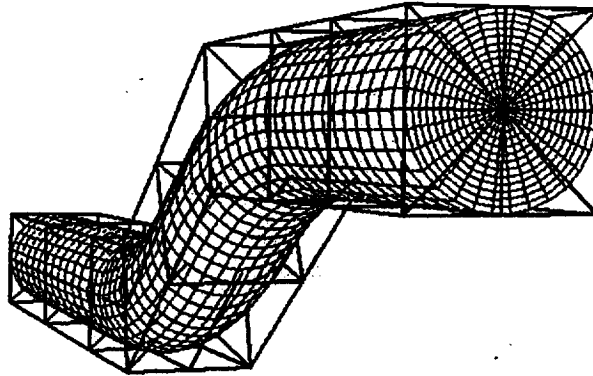

Figure 4.15 A NURBS volume grid with a singular plane.
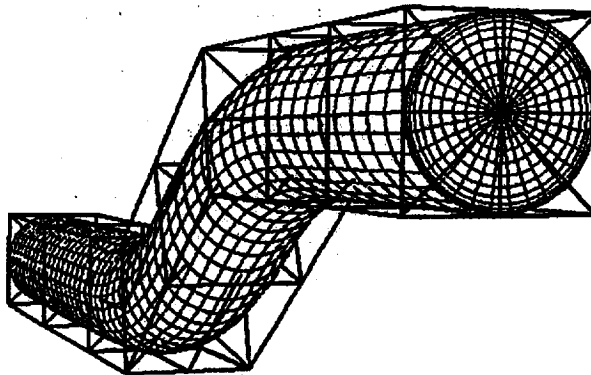
Figure 4.16 Reparameterization for a NURBS volume with singularity.

Efficient Evaluation for NURBS Entity

As it was described in the beginning of this chapter, using the intrinsic computer recursive call for evaluating the NURBS entity is the worst choice. Currently, the "de Boor" algorithm [Ref 18,23], which is based on the geometric construction, is widely utilized. The illustration of this algorithm is demonstrated with a BSpline curve, which has the same curve configuration as shown in the Figure 4.17.
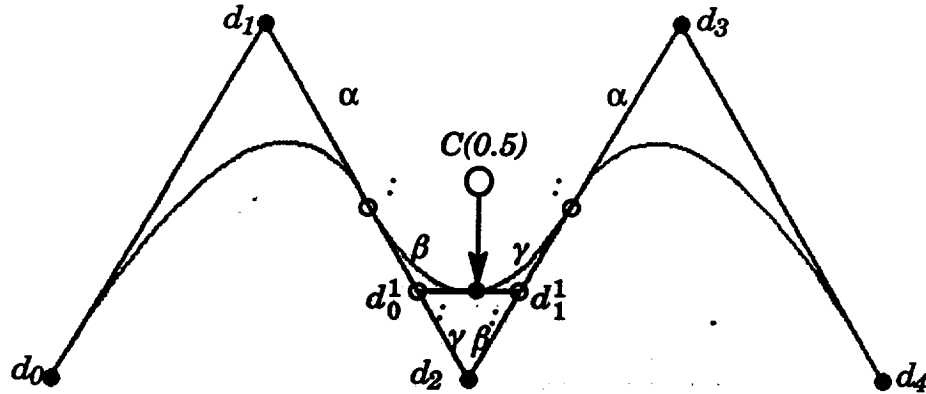
Figure 4.17 Illustration of the de Boor algorithm for a NURBS curve.

Before discussing the evaluation of a NURBS curve, the BSpline curve should be introduced first. The definition of the BSpline curve is similar to the NURBS curve except the *weights* are all equal to 1. According to the property of the basis function, if all the weights are the same, the denominator term of the equation (2.1) will sum to 1. Hence, the BSpline curve is presented as equation (4.3).

$$C(s) = \sum_{i=0}^{m} d_i \, N_i^k(s) \qquad (4.3)$$

If one would like to evaluate $t = 0.5$ for this curve, the first step is locating the proper knot span such that $T_j \leq t < T_{j+1}$ for the knot vector T. For this

case, the I would be 3 since the knot vector is set as [0, 0, 0, 0.3333, 0.6667, 1, 1, 1]. The second step is mapping the ratios of $T_2 : T_3$, $T_3 : t$, $t : T_4$ (let these ratios be α, β and γ) and the ratios of $T_3 : t$, $t : T_4$, $T_4 : T_5$ (β, γ and α) to the proper control polygon shown in Figure 4.17. This mapping will yield two intermediate points d_0^1, d_1^1. The third step is mapping the ratio of $T_3 : t$ and $t :$ T_4 to the line of $d_0^1 d_1^1$. The intermediate point created after this step is the point on the curve at parametric t. These procedures are demonstrated on Figure 4.17.

The algorithm for obtaining all the intermediate points and the final point on a BSpline curve (with order equal to k) for a particular parametric value t can be summarized as *Algorithm 4.7*.

Algorithm 4.7

> find out the index I such that $T[I] \leq t < T[I+1]$.
> for (r=1; r $\leq (k-1)$; r++)
> > for (i= $I - (k-1)$; i $\leq (I-r)$; i++)
> > $$d_i^r = \frac{(T[i + k] - t)}{(T[i + k] - T[i + r])} \, d_i^{r-1} + \frac{(t - T[i + r])}{(T[i + k] - T[i + r])} \, d_{i+1}^{r-1}$$

The evaluation of a NURBS curve by the "de Boor" method is related to "homogenization". The 4D control vertices $d^* = [\ d_x w, \ d_y w, \ d_z w, \ w\ ]$ are obtained by multiplying the *weights* to the corresponding 3D control polygon [d_x, d_y, d_z], and these new control vertices d^* (including the weights) are called the 4D homogenous coordinates. Performing the "de Boor" algorithm to these 4D control polygons for the particulate parametric value t will yield a new 4D point $C^*(t) = [\ d_{ix}\ w_i, \ d_{iy}\ w_i, \ d_{iz}\ w_i, \ w_i\ ]$. The corresponding point of $C^*(t)$ on 3D space is then obtained by "inhomogenizing" the 4D point and is accom-

plished by dividing the $C^*(t)$ with the last component, w_t. Hence, the 3D point $C(t) = [d_{ix}(t), d_{iy}(t), d_{iz}(t)]$.

The definition of the BSpline surface is similar to NURBS definition. The only different is the *weights* of the BSpline surface are all same. This leads the denominator term of equation (2.3) vanish [Ref 34,44]. Hence, the formula is represented as equation (4.4).

$$S(s,t) = \sum_{j=0}^{n} \sum_{i=0}^{m} d_{ij}\, N_i^{k1}(s)\, N_j^{k2}(t) \tag{4.4}$$

Since the BSpline surface is also defined as a tensor product surface, the equation (4.4) can be rewritten as equation (4.5).

$$\sum_{j=0}^{n}\left[\sum_{i=0}^{m} d_{ij}\, N_i^{k1}(s)\right] N_j^{k2}(t) = \sum_{j=0}^{n} d_j^*\, N_j^{k2}(t) \tag{4.5}$$

This equation shows that the evaluation of a BSpline surface can be accomplished by applying the BSpline curve algorithm to the isoparametric curves in I direction to obtain the intermediate control points, and then treat them as the isoparametric control polygon in J direction, applying the curve algorithm to these control polygons to obtain the final BSpline surface. In other words, the surface algorithm is implemented by applying two of the curve algorithms.

The detailed implementation of the NURBS surface can be found in Farian's [Ref 23,34,44]. Basically, one should apply the "homogenization" to obtain the 4D control vertices $d_{ij}^* = [dx_{ij}w_{ij}, dy_{ij}w_{ij}, dz_{ij}w_{ij}, w_{ij}]$, then treat these new control vertices as the control points of a BSpline surface and perform the BSpline evaluation algorithm in 4D space for the particular parametric values. Finally, project the values back to 3D space by "inhomogenizing".

The evaluation of NURBS volumes by "de Boor" algorithm can be extended analogously. Compared to the method using the recursive call implemented in many computer languages, this approach avoids the recursive call and imbeds the evaluation of the basis functions to the entire process. The evaluation process is much faster. Also, the geometric construction process helps the user to know how the points are generated. Many software packages adopt this algorithm. For example, the *NGP* (National Grid Project) [Ref 29,71] used this approach to evaluate the NURBS entities.

Even though the computation of the "homogenizing" and "inhomogenizing" of the "de Boor" algorithm is not very expensive, it is an extra cost for the entire process. Also, the algorithm requires the iterative process repeated for evaluating the x, y, z and w components. Examine the equations (2.1) ~ (2.4) thoughtfully. One should know that after the basis functions are evaluated, the control points regarding the x, y, z components can share these basis functions without repeated calculation. The detail approach is described as follows:

Consider all the basis functions of the NURBS curve shown in Figure 4.9. According to equation (2.2), those basis functions can be plotted as shown in Figure 4.18.
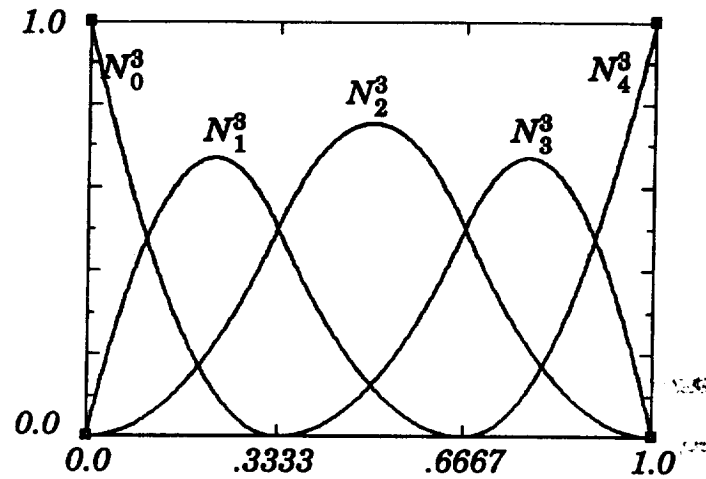
Figure 4.18 Non–zero quadratic basis functions.

From Figure 4.18, the first basis function exists in the knot span [0.0, .3333], and second basis function exists in the knot span of [0.0, .6667], and so forth. Also, notice that in any of the knot span, the non–zero basis functions sum to 1. For example, for parametric value t = .3333, the basis functions $N_1^3(t)$ + $N_2^3(t)$ + $N_3^3(t)$ = 0.5 + 0.5 + 0.0 = 1.0. This situation also indicates that the number of non–zero basis functions in any of the particular knot span is equal to the *order* of the curve (3 for this case). (Even though $N_3^3(t)$ is zero at t = 0.3333, but as t slightly increases, this basis function becomes non–zero.) Due to the local control property, one can also understand that not all the control points are involved for a particular t, only 3 (equal to the *order* of this NURBS curve) control points are used for the evaluation. For example, if the parametric value t is located in the knot span [0.0, .3333], the control points of d_0, d_1 and d_2 are used, if it is in the knot span [.3333, .6667], the control points of d_1, d_2 and d_3 are used, if it is in the knot span [.6667, 1.0], then the control points of d_2, d_3 and d_4 are used. Generally, if the parametric value t is located in the knot span of [T_I, T_{I+1}], then the control points of $d_{I-(order-1)}$,

$d_{I-(order-1)+1}$, ..., d_I are involved for the evaluation. This statement can also be applied to the NURBS surface and volume. Take the surface as an example: if the parametric values (s, t) are within the criteria of $T_1[I] \leq s < T_1[I+1]$ and $T_2[J] \leq t < T_2[J+1]$, then the control net involved for the evaluation is in the region of $d_{I+i,J+j}$ where $i = I - (order\_i - 1)$, ..., I $j = J - (order\_j - 1)$, ..., J. The volume can be extended analogously.

With this knowledge, the evaluation for the NURBS entity is fairly straightforward. If the user would like to evaluate a NURBS curve for ni points with parametric values t_i $i=0$, ... $ni-1$, then the evaluation procedure for this NURBS curve (with *order* k and the control polygon d_i $i=0$, ..., n) can be presented by *Algorithm 4.8*.

Algorithm 4.8

```
for (i=0; i<ni; i++) {
    I = locate the proper knot span for t[i];
    find all non-zero basis functions, B[0:k-1];
    var1 = var2 = var3 = var4 =0.0;
    for (j=0; j<k; j++) {
        var1 += w[j+I-(k-1)] × B[j];
        var2 += w[j+I-(k-1)] × dx[j+I-(k-1)]×B[j];
        var3 += w[j+I-(k-1)] × dy[j+I-(k-1)]×B[j];
        var4 += w[j+I-(k-1)] × dz[j+I-(k-1)]×B[j];
    }
    curx[i]=var2 / var1; cury[i]=var3 / var1; curz[i]=var4 / var1;
}
```

From *Algorithm* 4.8, one can understand that when the non-zero basis functions are obtained for the parametric value t_i in the particular knot span, then only k (*order* of the curve) operations are needed to evaluate the point of $C(t_i)$. Compare to the "de Boor" geometric construction, this algorithm is more intuitive and easy to understand based on the equation (2.1).

The other advantage of this approach is that it is fairly easy to extend from the 1D curve case to higher dimensions. Since the NURBS surface and volume are all defined by the tensor product form, the corresponding algorithms for the surface and volume are shown as *Algorithm* 4.9 and 4.10, respectively.

The NURBS information (the *order*(s), the control vertices, the *weights*, and the knot vector(s)) is all given in the *Algorithm* 4.8 ~ 4.10, the only challenge left for a fast computation is obtaining all the non–zero basis functions efficiently in the particular knot span. Indeed, the evaluation of the recursive defined basis functions is the key point of *Algorithms* 4.8 ~ 4.10.

Algorithm 4.9

```
for (j=0; j<nj; j++)
for (i=0; i<ni; i++) {
    I = locate the proper knot span for s[i][j];
    J = locate the proper knot span for t[i][j];
    find all non–zero basis functions, Bi[0:k1-1], Bj[0:k2-1];
    var1 = var2 = var3 = var4 =0.0;
    for (jj=0; jj<k2; jj++)
    for (ii=0; ii<k1; ii++) {
        var1+=w[ii+I-(k1-1)][jj+J-(k2-1)]×Bi[ii]×Bj[jj];
        var2+=w[ii+I-(k1-1)][jj+J-(k2-1)]×dx[ii+I-(k1-1)][jj+J-(k2-1)]×
                Bi[ii]×Bj[jj];
        var3+=w[ii+I-(k1-1)][jj+J-(k2-1)]×dy[ii+I-(k1-1)][jj+J-(k2-1)]×
                Bi[ii]×Bj[jj];
        var4+=w[ii+I-(k1-1)][jj+J-(k2-1)]×dz[ii+I-(k1-1)][jj+J-(k2-1)]×
        }       Bi[ii]×Bj[jj];
    surx[i][j]=var2 / var1; sury[i][j]=var3 / var1; surz[i][j]=var4 / var1;
}
```

Algorithm 4.10

```
for (l=0; l<nl; l++)
for (j=0; j<nj; j++)
for (i=0; i<ni; i++) {
    locate the knot span I, J, L for s[i][j][l], t[i][j][l], u[i][j][l];
    find all non-zero basis functions, Bᵢ[0:k₁-1], Bⱼ[0:k₂-1], Bₗ[0:k₃-1];
    var₁ = var₂ = var₃ = var₄ =0.0;
    for (ll=0; ll<k₃; ll++)
    for (jj=0; jj<k₂; jj++)
    for (ii=0; ii<k₁; ii++) {
```

$var_1 +=w[ii+I-(k_1-1)][jj+J-(k_2-1)][ll+K-(k_3-1)]\times B_i[ii]\times B_j[jj]\times B_l[ll];$

$var_2 +=w[ii+I-(k_1-1)][jj+J-(k_2-1)]\times B_i[ii]\times B_j[jj]\times B_l[ll]\times$
$\qquad dx[ii+I-(k_1-1)][jj+J-(k_2-1)][ll+L-(k_3-1)];$

$var_3 +=w[ii+I-(k_1-1)][jj+J-(k_2-1)]\times B_i[ii]\times B_j[jj]\times B_l[ll]\times$
$\qquad dy[ii+I-(k_1-1)][jj+J-(k_2-1)][ll+L-(k_3-1)];$

$var_4 +=w[ii+I-(k_1-1)][jj+J-(k_2-1)]\times B_i[ii]\times B_j[jj]\times B_l[ll]\times$
$\qquad dz[ii+I-(k_1-1)][jj+J-(k_2-1)]\times[ll+L-(k_3-1)];$

```
    }
```

$surx[i][j]=var_2/var_1;$ $sury[i][j]=var_3/var_1;$ $surz[i][j]=var_4/var_1;$

```
}
```

An efficient evaluation of the basis functions was first presented in de Boor's [Ref 18]. This approach was implemented by an iterative (non–recursive) approach. This evaluation process is briefly discussed as follows:

Based on the previous observation, one knows that for a NURBS curve with *order* equal to k, if the parametric value t is located in the knot span I such that $T_I \leq t < T_{I+1}$, then there are at most k non–zero basis functions. Also, according to equation (2.2), one knows a basis function of k *order* is defined by the combination of two basis functions with $k-1$ *order*. These two facts conclude to the relationship, shown as Figure 4.19, of all the non–zero basis functions with all possible *orders* for the parametric value t in the knot span I.
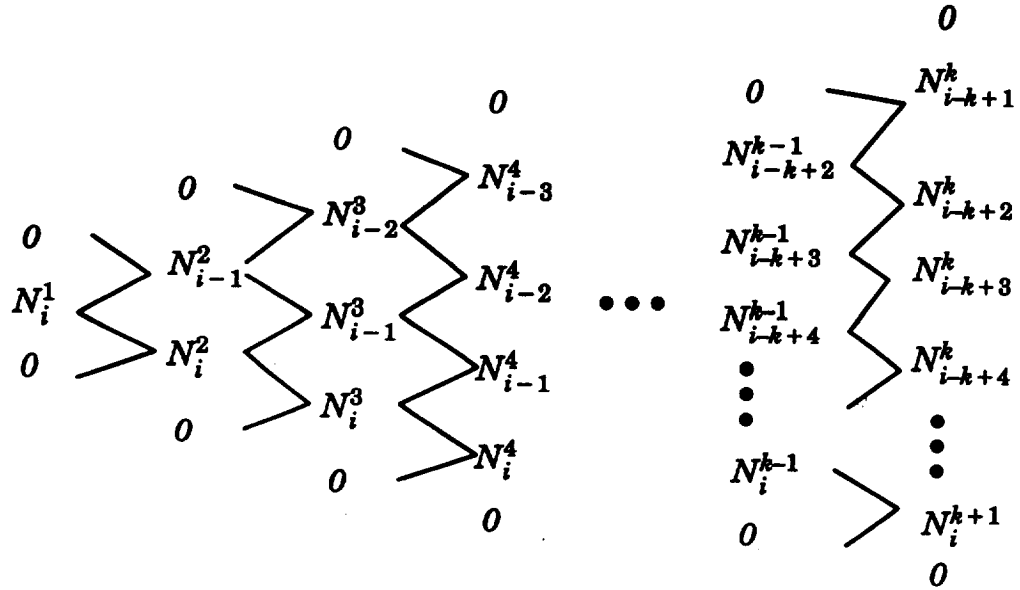
Figure 4.19 Nonzero BSpline basis functions on $[T_I, T_{I+1}]$

In each column of Figure 4.19, there are exactly k non–zero basis functions, and any basis function in the jth column is obtained by the combination of two other basis functions in the $j-1$th column. Also, notice that when computed the basis functions N_i^j and N_{i-j+1}^j in the jth column, the fact that one of their neighbors to the $j-1$th column is zero is utilized. For example, when calculating N_i^2, the N_{i-1}^1 is zero, and when calculating N_{i-1}^2, the N_{i+1}^1 is zero. In addition to this, there exists a simple relationship between all the non–zero basis functions in the jth column. Take j equals 3 (order equal to 3) as an example, according to equation (2.2), these non–zero basis functions can be listed as follows:

$$N_{i-2}^3(t) = \frac{(t - T_{i-2})}{T_i - T_{i-2}} N_{i-2}^2(t) + \frac{(T_{i+1} - t)}{T_{i+1} - T_{i-1}} N_{i-1}^2(t) \qquad (4.6)$$

$$N_{i-1}^3(t) = \frac{(t - T_{i-1})}{T_{i+1} - T_{i-1}} N_{i-1}^2(t) + \frac{(T_{i+2} - t)}{T_{i+2} - T_i} N_i^2(t) \qquad (4.7)$$

$$N_i^3(t) = \frac{(t - T_i)}{T_{i+2} - T_i} \ N_i^2(t) + \frac{(T_{i+3} - t)}{T_{i+3} - T_{i+1}} N_{i+1}^2(t) \tag{4.8}$$

The relationship is that the $\dfrac{1}{T_{i+1} - T_{i-1}} N_{i-1}^2$ which appears in the second term of equation (4.6) also appears in the first term of equation (4.7). Similar results also can be applied to equations (4.7) and (4.8). Furthermore, two qualities are introduced and shown as equation (4.9) and (4.10).

$$\delta\_left(j) = t - T_{i+1-j} \tag{4.9}$$

$$\delta\_right(j) = T_{i+j} - t \tag{4.10}$$

This simple relationship can be presented in a clearing fashion by applying equations (4.9) ~ (4.10) to equations (4.6) ~ (4.8). The re–formed equations are shown as equations (4.11) ~ (4.13).

$$N_{i-2}^3 = \frac{\delta\_left(3)}{\delta\_right(0) + \delta\_left(3)} N_{i-2}^2 + \frac{\delta\_right(1))}{\delta\_right(1) + \delta\_left(2)} N_{i-1}^2 \tag{4.11}$$

$$N_{i-1}^3 = \frac{\delta\_left(2)}{\delta\_right(1) + \delta\_left(2)} N_{i-1}^2 + \frac{\delta\_right(2)}{\delta\_right(2) + \delta\_left(1)} N_i^2 \tag{4.12}$$

$$N_i^3 = \frac{\delta\_left(1)}{\delta\_right(2) + \delta\_left(1)} N_i^2 + \frac{\delta\_right(3)}{\delta\_right(3) + \delta\_left(0)} N_{i+1}^2 \tag{4.13}$$

Equations (4.11) ~ (4.13) provide the intrinsic understanding for an efficient evaluation of all non–zero basis functions for a parametric value t in the knot span of $[T_i, T_{i+1}]$. This efficient evaluation process, which utilizes an iterative approach, for these non–zero basis functions is listed as *Algorithm 4.11*.

Algorithm 4.11

```
basis[0] = 1.0;
for (j=0; j< order; j++) {
        delta_l[j] = t − knot[I+1−j];
        delta_r[j] = knot[I+j] − t;
        save = 0.0;
        for (i=0; i<j; i++) {
                term = basis[i] / ( delta_r [i+1] + delta_l[j−i];
                basis[i] = save + delta_r [i+1] * term;
                save = delta_l [j−i] * term;
        }
        basis[j] = save;
}
```

Currently, *CAGI* (*Computer Aided Grid Interface*) applies *Algorithm 4.11* to *Algorithms* 4.8 ~ 4.10 for evaluating the NURBS curves, surfaces and volumes.

Compared to the different evaluation approaches discussed in this section, the recursive algorithm needs the most computational time. The geometric construction of "de Boor" method is currently the one used most for many packages [Ref 21,28,44]. However, the one adopted in *CAGI* is the most efficient algorithm — both in memory and computation time. The reasons for this is that it avoids the processes of "homogenization", "inhomogenization", and also when the non−zero basis functions are obtained, these non−zero functions can be applied to any coordinates (x, y and z) without repeating the same iterations. Besides these advantages, it is the easiest to extend to any higher dimensions. In order to compare the computation speed for different evaluating approaches, a *C* program is implemented for the test. This program is executed in a personal IRIS machine which has "R2000A/R3000" processor chip, 56 Mbytes main memory and 64 Kbytes of cache. Let the algorithms

adopted in *CAGI* be denoted as algorithm *A*, and the "de Boor" be denoted as algorithm *B*, Tables 4.1 and 4.2 show the CPU time required for algorithms *A* and *B* for different NURBS curves and surfaces.

Table 4.1 Comparison time for evaluation of NURBS curves.

order (k)	resolution (np)	control point (n)	CPU time for A (microseconds)	CPU time for B (microseconds)
4	1000	5	49333	128000
4	5000	5	248000	650000
6	1000	18	93000	293000
6	5000	18	464500	1478250

Table 4.2 Comparison time for evaluation of NURBS surfaces.

orders (k1,k2)	resolutions (ni, nj)	control net (m, n)	CPU time for A (seconds)	CPU time for B (seconds)
4, 3	100, 100	5, 3	0.4650	0. 5710
4, 3	500, 500	5, 3	11.719	18.509
12, 10	100, 100	79, 230	3.7690	46.398
12, 10	500, 500	79, 230	92.739	562.068

The command *"getrusage"* was used to measure the *CPU* time needed. The *CPU* time was obtained by repeating the same case for 3 times and then averaging the total time spent.

From the results shown in Table 4.1 and 4.2, one can observe that the two methods are very completive when the *orders*, resolution and control points are low. However, the *CPU* time needed for algorithm *A* is much smaller than that of algorithm *B* while the *orders*, resolutions and control net are highly increased.

Remark

As is aforementioned, the NURBS representation is becomeing the standard for geometry description in CAD/CAM design systems. In addition, because of its properties, NURBS has been used for modeling objects in many areas, such as the manufacturing industry, entertainment industries, art and even modeling the scenes for virtual reality applications. In terms of numerical grid generation and CFS applications, the difficulties and barriers of using NURBS must be solved to encourage the CFS community to utilize this modeling technique. These problems discussed in this chapter are important. If the volume grids generated by NURBS can not maintain the geometry fidelity, then the CFS algorithm simply solves the different (or wrong) problems. The distribution control on grid points and the bad parameterization problems are highly related to grid quality — which has a direct influence on the flow convergence rate and solution precision. A grid with bad quality may even lead to the divergence of the CFS simulation results. Even though the computational ability has been improved for many different types of computers, a fast, robust and efficient evaluation for NURBS is required to facilitate the grid generation process. This is especially true when handling high resolutions moving grids (or moving boundary problems) which require fast evaluation in each time step.

CHAPTER V

NURBS IN DYNAMIC GRID GENERATION

After discussing the difficulties of utilizing NURBS and the associated overcoming strategies, one can fully apply the NURBS to many CFD simulations. As is discussed in previous chapters, the reparameterization algorithm allows the user to control the distribution of the grid points (lines) in physical space. This is useful for the surface (volume) grid refinement, particularly for the application of grid adaptation. The approach used here is different than that in Yang's [Ref 83,84]. In Yang's work, the BSpline interpolation is used for the iso—curve interpolation, while in this application, the entire distribution (adaptive) mesh / volume is utilized for the new adaptive grids. The interpolation involves the accumulation of numerical errors which may lead to the losing of the original geometry definition, especially for the viscous turbulent flow simulations. In addition, this interpolation is utilized for the entire surface grid by iso—curve interpolation, it will be a computational intensive work if the resolution of the surface grid is high. The approach used in this application will avoid these challenges. The computational examples are presented in this chapter.

An efficient grid generation with a good quality grid is important for most CFS simulations. This is particularly true for the unsteady deforming geometry type of simulation. For this type of problem, there is a need to generate the grid at each time step. If the grid generation is not efficient, a significant part of the computation time will be used for generating the grid. Also

if the grid quality is not satisfactory, the final results may not be accurate, or it may take more time for convergence. Hence, the grid generation procedure plays an important role in this type of simulation. NURBS is a good candidate for generating grids for this type of problem. The reasons for this are the parametric formula of NURBS can easily rebuild the geometry, and the robust evaluation algorithm described in the previous chapter makes the generating grid efficient. Many NURBS geometries are constructed by very concise control polygons (control net or control volume). This means that comparing the grid sizes, the size of the NURBS control net is much smaller than that of the entire grids. And the NURBS local control property allows the geometry to be locally modified without altering the entire geometric shape. If there is a need to create a local deforming geometry, it is ideal to perform the deformation to the NURBS control points. Since the resolution of the NURBS control net is smaller, it does not take much computational time to apply perturbations to the geometry. In addition, the reparameterization algorithm provides a desired distribution on the deformed geometries to maintain the grid quality at different time steps. Also, the grid smoothness can be achieved by the combination of the proper NURBS *orders* and knot vectors. This is because the continuity of the NURBS entity is determined by the associated *order* and knot vector and is equal to $C^{(k-1-m)}$ where k is the *order* and m is the multiplicity of the evaluated knot value. These applications can be found in the moving grid example and the temporally deforming geometries. These issues are discussed as follows:

Grid Adaptation

Since the re–parameterization algorithm can control the grid packing on the geometry in physical space, its application can then be used for the grid

adaptation. This is demonstrated by Figures 5.1 ~ 5.5 which show a 2D NURBS surface grid modeled for simulating a supersonic flow over an oval–type object. A NURBS control net is generated to model the 2D flow field configuration. As it was discussed in chapter 2, a circular arc can be represented by NURBS with very few control polygons. For a semi–circular arc with a radius r, as shown in Figure 5.1, the 2D control points are (counterclockwise) $(r,0)$, (r,r), $(0,r)$, $(-r,r)$ and $(-r,0)$, while the corresponding *weights* are 1, $\cos(45^0)$, 1, $\cos(45^0)$ and 1. The knot values are (0, 0, 0, 0.5, 0.5, 1, 1, 1) with the *order* set to 3. In order to simulate the flow passing through the oval–type object, one can apply the algorithm described in Piegl's [Ref 42,43,44] for the semi–ellipse. However, the easiest and simplest way to model this conic arc (the ellipse) is by utilizing the "scaling" operation: since this r is not fixed, another semi–circular arc can be created with radius r' such that $r' < r$ while keeping the *weights*, knot values and the *order* unchanged, then "scaling" the entire control polygon by scaling the x and y components of each control point by $(a/r, b/r)$ — yielding a new control polygon $(a,0)$, (a,b), $(0,b)$, $(-a,b)$ and $(-a,0)$ of the semi–ellipse, hence, the semi major is a and semi minor is b. After these two NURBS curves are created, one can perform the "NURBS Ruled" surface algorithm (described in Chapter Two) to generate the surface grid for the flow field simulation. The NURBS information of this surface is listed in Appendix B for reference. Supersonic flow at Mach number 3.0 is applied to this geometry. After the initial grid is generated, the NPARC flow simulation code [Ref 16] is used to create the initial solution. These examples are demonstrated in Figures 5.2 and 5.3.
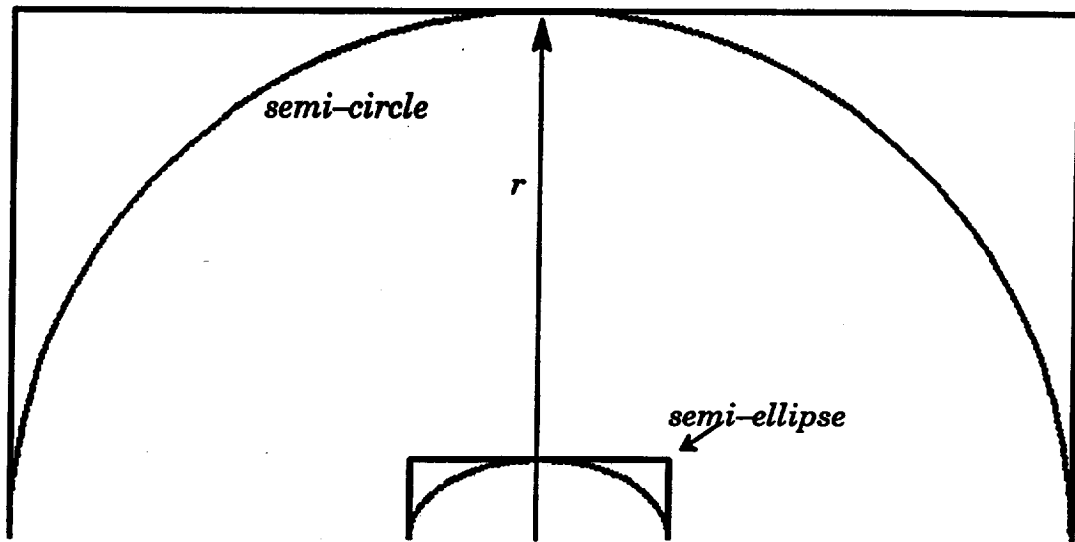
Figure 5.1 NURBS curves for semi–circle and ellipse.
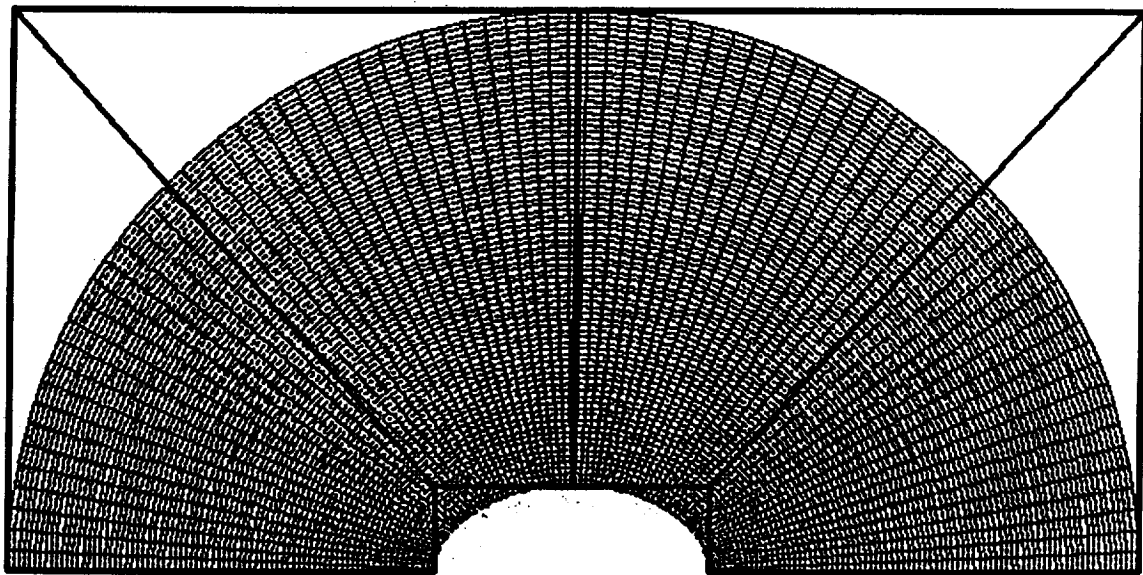
Figure 5.2 NURBS control net with the initial surface grid.

0.00   2.16   4.31
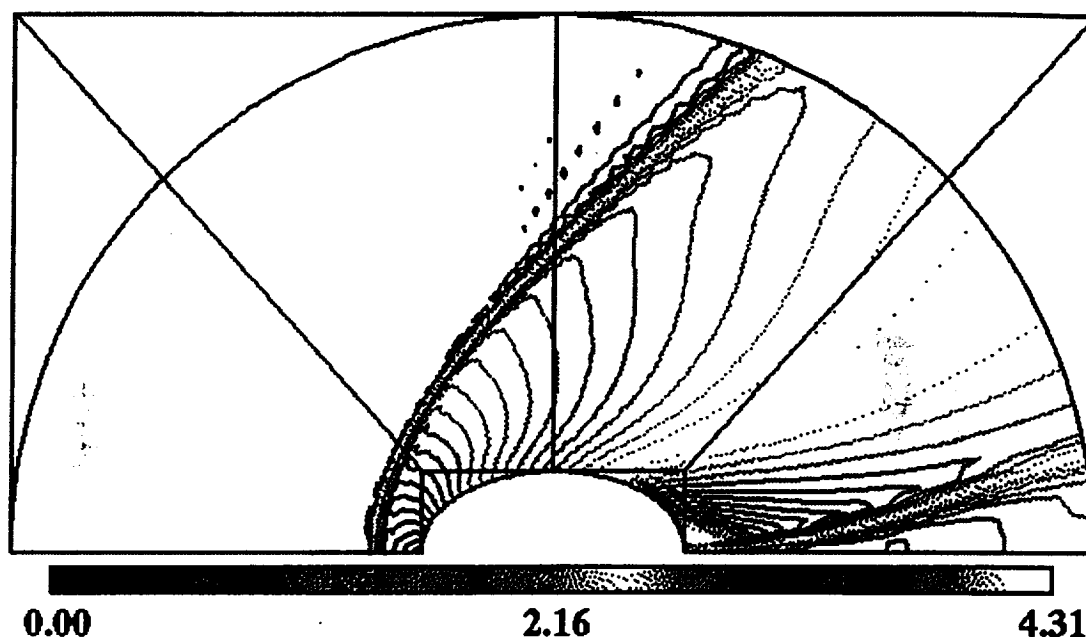
Figure 5.3 Initial solution (Mach number plot).

After obtaining the initial grid and solution, the adaptive program *"GAGE2D"* by Yang [Ref 83,85] is used for calculation of the distribution mesh and the weight functions. *GAGE2D* was modified, however, with this new approach for full NURBS evaluation. The idea is that after the adaptive code calculates the adaptive mesh, instead of interpolating iso–curve line by line, the re–parameterization algorithm is utilized to generate a new surface grid which can reflect the adaptive mesh to the physical space. Figures 5.4 abd 5.5 show the adaptive grids (with NURBS control net) and the corresponding solutions after the adaptive iterations.

Similar to this 2D case, a 3D NURBS control volume is used to model a missile for flow simulation. The 3D NURBS control volume and the initial grid for this generic missile configuration are shown in Figure 5.6. This control volume is generated by creating two NURBS surfaces – one for the missile

Figure 5.4 NURBS adaptive grid (first iteration).

0.00   2.12   4.24

Figure 5.5 Adaptive solution (first iteration, Mach number plot).

configuration and the other for the outer free stream surface. The "ruled volume" algorithm is then applied to these two NURBS surfaces for constructing the final control volume. Supersonic flow at Mach = 2.5 and 14 degree angle of attack is applied to this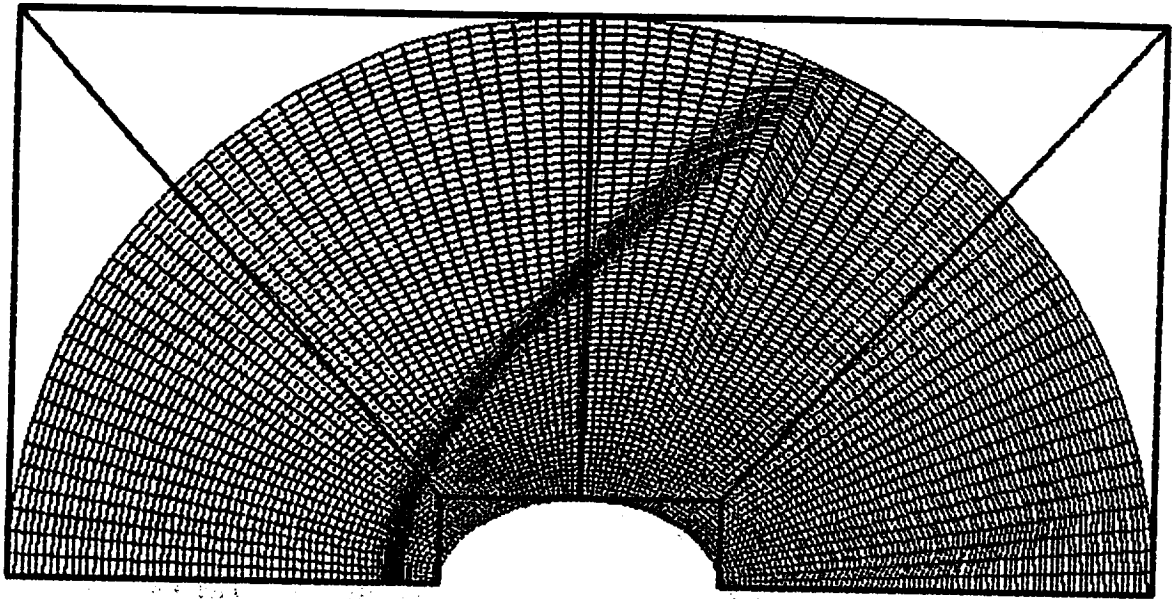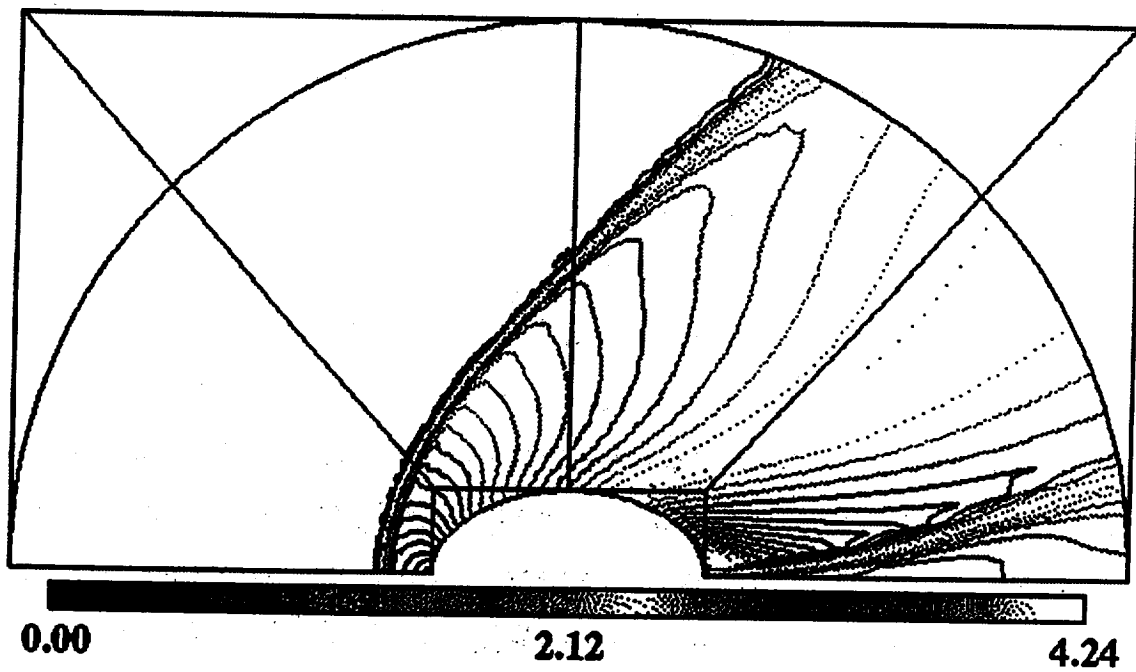 geometry. Because of the simulation of the angle of attack, there is no need to generate a symmetric grid around the body. Hence, the NURBS control volume should not be symmetric in the circumferential direction. The outer free stream NURBS surface was originally created from a surface of revolution algorithm. In order to model a non-symmetric surface, half of the control net must be reduced by a scaling factor. This, however, results in the initial grid with the uneven distribution in circumferential direction (K direction) even though the even parametric values are used. One should be able to observe this "bad parameterization" situation from Figure 5.6.

Figure 5.7 presents the flow solution obtained from NPARC [Ref 16] flow solver with the *Baldwin-Lomax* algebraic turbulence model option. This Figure shows a streamwise cut of the initial grid and solution. The grid size for this simulation is set as 121 × 89 × 89.

Similar to the 2D adaptation case, the adaptive volume (the new parametric values) is calculated according to the the initial solution. Unlike the approach of line by line interpolation used for obtaining the adaptive grid in Yang's [Ref 83,84] and Thornburg's [Ref 67], the 3D re-parameterization algorithm is utilized, along with this new adaptive volume, to obtain the final adaptive volume grid. Figure 5.8 shows this adaptive volume grid in the perspective view. This re-parameterization technique provides the capability of precise grid distribution control and also maintains the geometry fidelity. One can notice that the "bad parametrization" shown in Figure 5.6 has been fixed
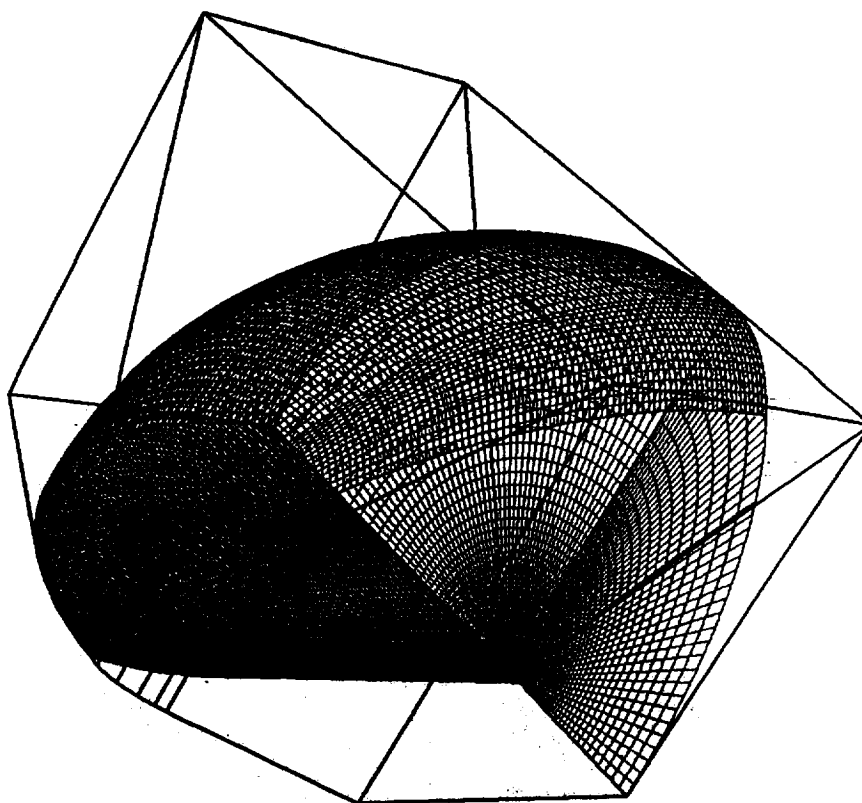
Figure 5.6 NURBS control volume for a generic missile configuration.
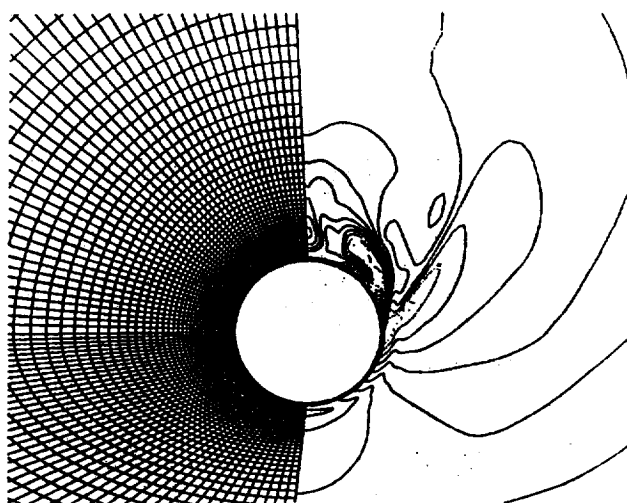
Figure 5.7 Initial grid and solution (Mach number plot).

after the adaptive procedure. The adaptive grid and solution in streamwise cut are shown in Figure 5.9. Figure 5.10 presents the adaptive solution with the NURBS control volume in the perspective view.
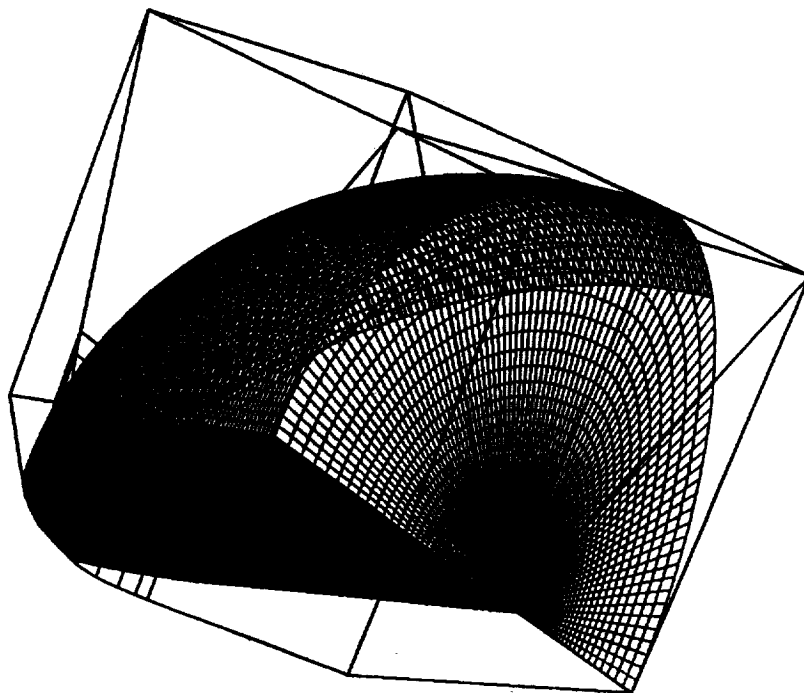
Figure 5.8 Adaptive grid in perspective view.
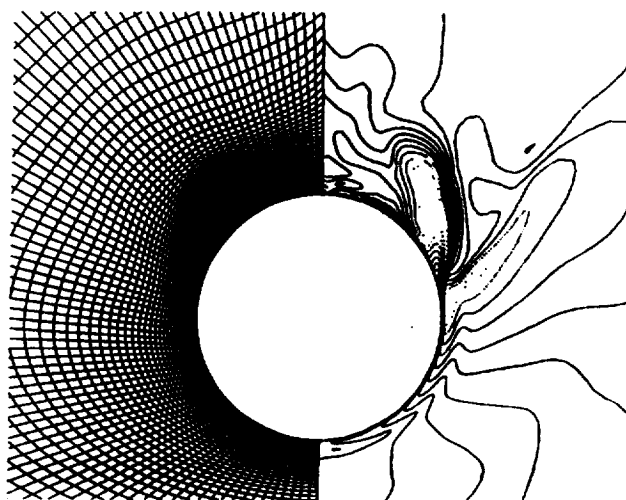
Figure 5.9 Adaptive grid and solution (Mach number plot).
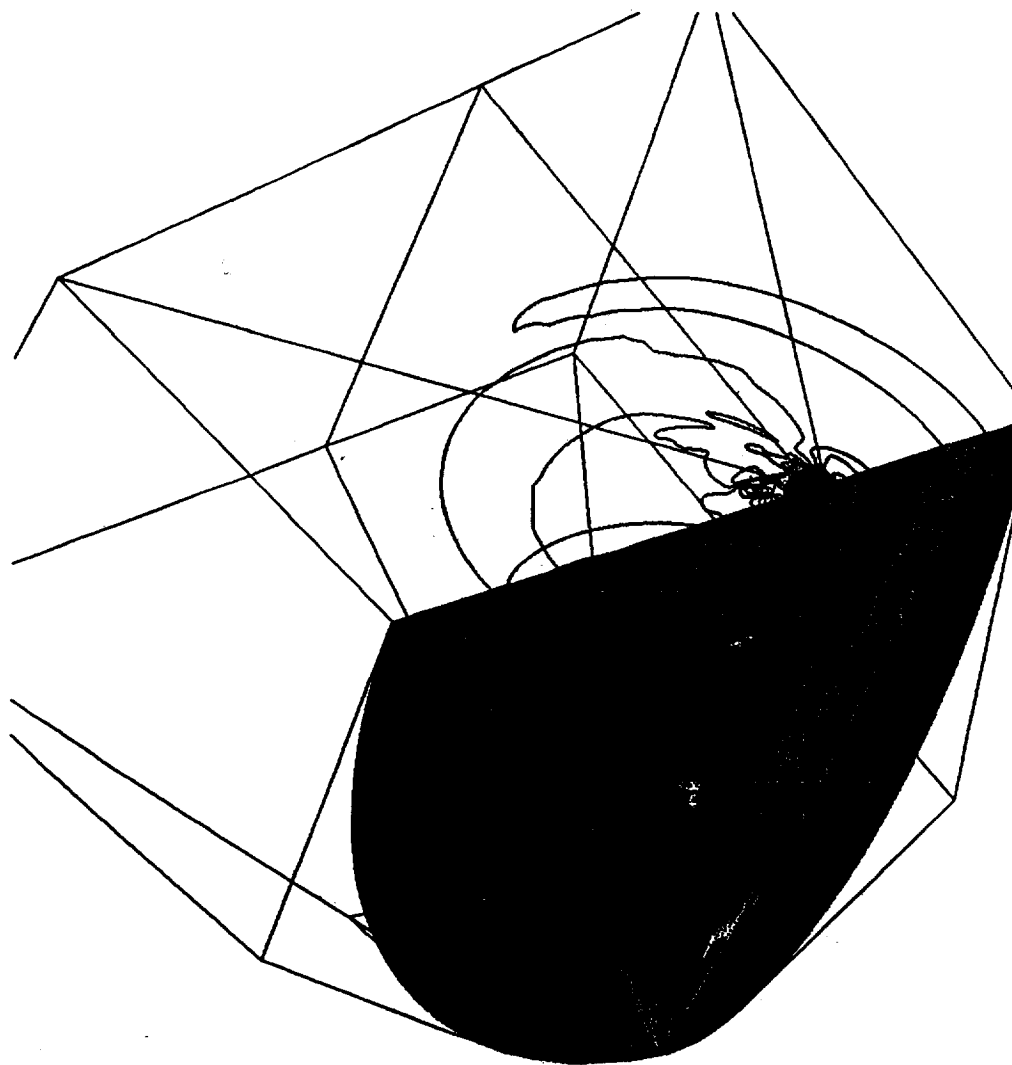
Figure 5.10 Adaptive solution in perspective view.

Temporally Deforming Geometry Model by NURBS

Unlike the other geometric representation, NURBS allows the engineer to modify the geometry locally by changing the associated control points or the associated *weights*. This property is called the "local control" property. The other geometric representations, such as the Bezier and cubic spline entities, do not have the similar characteristics. Hence, if one changes the control points of the Bezier curve or the coefficients of a cubic spline surface, the entire geometry will change. This NURBS local control property attracts many designers because it provides the flexibility to model the geometric shape easily. Figure 5.11 demonstrates this property. This figure shows the original NURBS surface and the NURBS surface with the perturbed control point. One can observe that after the control point is perturbed, only the local surface is effected.
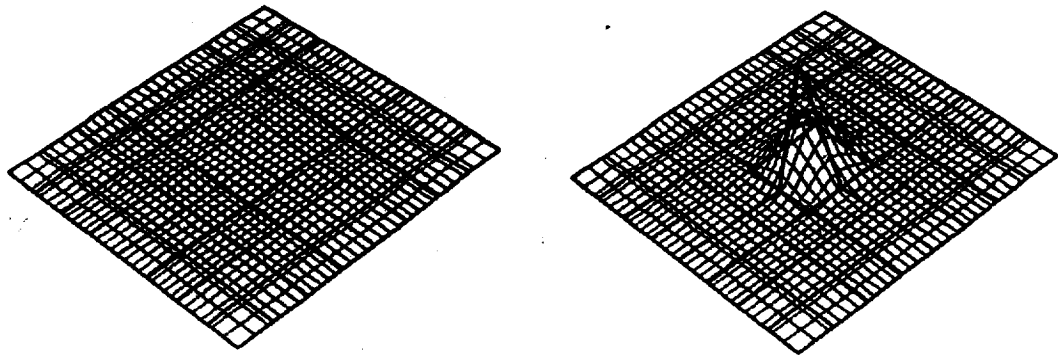
Figure 5.11 NURBS local control property.

NURBS can model some physical phenomenons by utilizing this local control property. For instance, by properly changing the location of the control polygons, the NURBS can be used to model the wave propagation. As is demonstrated in previous chapters, many of the geometries modelled by NURBS

have a smaller size of control polygon (net or volume) than that of the associated grid points. Manipulating the control polygon (control net or volume) instead of the grid points on curve (surface or volume) will save the computational time. It is reasonable to work on the control polygon and then re-evaluate the entire geometry with the efficient NURBS evaluation algorithm (described in chapter four). This strategy has been applied to CFD simulations, such as the moving boundaries problem or the temporally deforming geometry [Ref 13]. For the unsteady problems, the grids needed to be generated efficiently in each time step to shorten the simulation procedure. Manipulating the control polygon and then evaluating the grid will meet this requirement. The following examples show the deforming geometry problems with this approach.

NURBS Models the One Dimensional Wave Movement

A straight line can be easily modelled by NURBS with two control points and set the *order* equal to two. However, to model the wave propagation along a one dimensional curve, the *order* of the NURBS curve must be set to at least three (or greater than three) to get a smooth wave movement. Also, the approach presented in this study is applying the manipulation to the control polygon, hence, one has to set enough numbers of control points to catch enough simulating information. In this one dimensional case, a NURBS curve with nine control points is used. The *order* of this curve is three. In order to simulate the wave propagation along this curve, the one dimensional wave equation [Ref 1] is solved and applied to the associated NURBS control polygon at different time steps. The one dimensional wave equation can be described in equation (5.1) shown as follows.

$$u_t + au_x = 0 \qquad\qquad (5.1)$$

where u is the displacement in y axis direction, and a is the propagation speed. This simple wave equation can be solved by many available finite difference schemes [Ref 1]. However, in this study, instead of solving this equation to all the grid points (121 points) on the curve, the wave equation is applying to the NURBS control polygon. Originally, the control polygon (consisting of nine control points) lines in the x axis. At the first time step, the initial condition is set so that the y coordinate of the first control point is perturbed. The boundary condition is set to the left hand side of the curve. The y coordinate of the first control point is oscillating with certain damping functions at each time step. The explicit upwind scheme [Ref 1] is used to solve the equation. At each time step, when the new control points are obtained according to equation (5.1), the NURBS curve evaluation algorithm (described in Chapter four) is utilized to generate the new curve. Solving equation (5.1) will result in a propagating wave based on the initial condition and boundary condition. Hence, the final curve evaluated with different control polygons will result in the new curve which simulates the wave propagation. This simulation is demonstrated in Figure 5.12 at different time steps.

One may doubt why this approach is needed for this simulation, because many other approaches can reach the same result without using NURBS representation. The reason is that the proof of the versatility of using NURBS for different applications. And for the complicated cases, such as the surface or the volume with large grid points, this approach can reach the goal with less computational time.

time = 0 sec

time = 1 sec

time = 2 sec

time = 3.5 sec

time = 5 sec

NURBS control polygon.   NURBS curve.
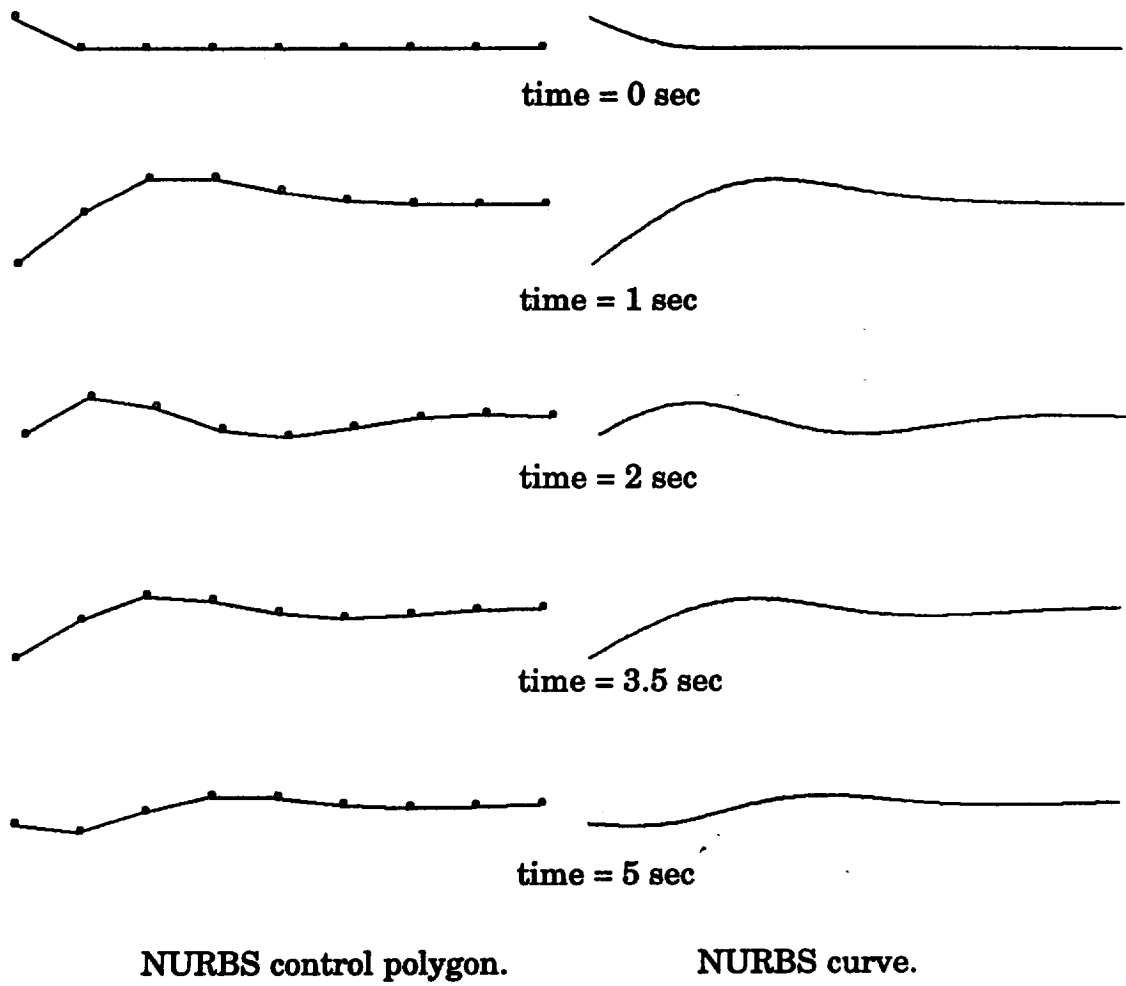
Figure 5.12 NURBS curve simulates the wave propagation.

NURBS Models the Two Dimensional Wave Movement

Similar to the one dimensional case, the NURBS surface can be used to simulate the two dimensional wave propagation. This is done by applying the two dimensional wave equation to the control net of the NURBS surface and then evaluating the surface grid with the new control nets at each time step. The resulting surface grid will be smooth if proper *orders* and knot vectors are selected for the NURBS evaluation. The 2D wave equation is described in equation (5.2) as follows.

$$u_t + au_x + bu_y = 0 \qquad (5.2)$$

A NURBS surface with *orders* 3 by 3 is used to model this wave propagation. The knot vectors are set so that the multiplicities of knot values in the evaluation domain are all equal to one. Hence, the surface grid will be C^1 continuously. Initially, the control net (consists of 19 by 19 control points) sits on the *xy* plane. At the time step equals to 0, the initial condition is set so that none of the control points is perturbed. The boundary condition is set to the center of the control net (the indices of the (9, 9) control point). The *z* coordinate of this control point is oscillating at each time step with a designed damping function. The function will create different *z* attitudes for the central control point at different time steps. The *z* displacement verses time steps of control points (9,9) is shown in Figure 5.13.
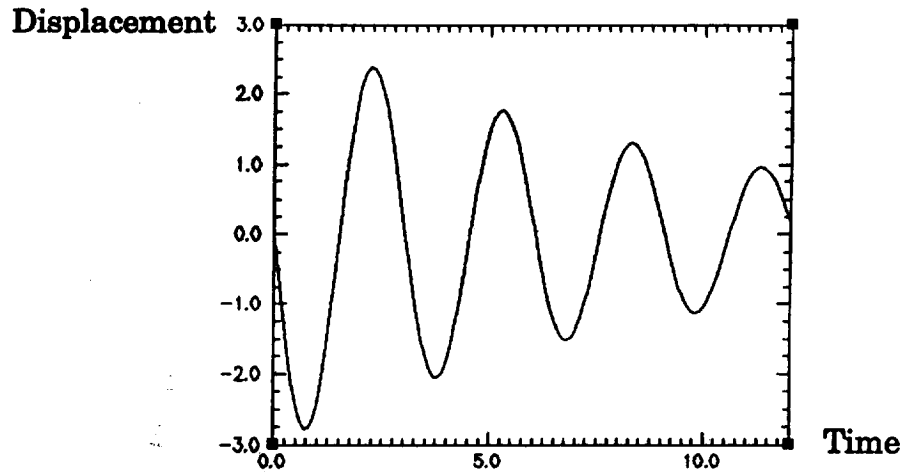
Figure 5.13 The oscillating function of the central control point.

An explicit finite difference scheme is used to solve equation (5.2) for propagating the z displacement. At each time step, when the new control net is obtained, the efficient evaluation algorithm (described in chapter four) is then used to generate the corresponding surface. In this simulation, the wave speeds (variables a, b in equation (5.2)) are set to be the same, and the time step is set to 0.1. This simulating process is demonstrated in the Figure 5.14 for different time steps.

The NURBS is composed of the control polygon (net or volume), *orders*, knot vectors and the *weights*. A previous example shows the deforming geometry obtained by manipulating the locations of the control points. Another alternative for modeling the motion grid is manipulating the *weights* of the NURBS representation. The next example of deforming geometry is demonstrated by manipulating not only the control points but also the associated *weights*. As is discussed in chapter two, the NURBS modeling algorithms for the circular arcs, elliptic arcs and the superelliptic arcs are analogous. For an arc (circular arc, elliptic arc or the superelliptic arc) with sector angle less

time = 0

time = 1 sec

time = 2 sec

time = 3 sec

NURBS control net.   NURBS surface grid.
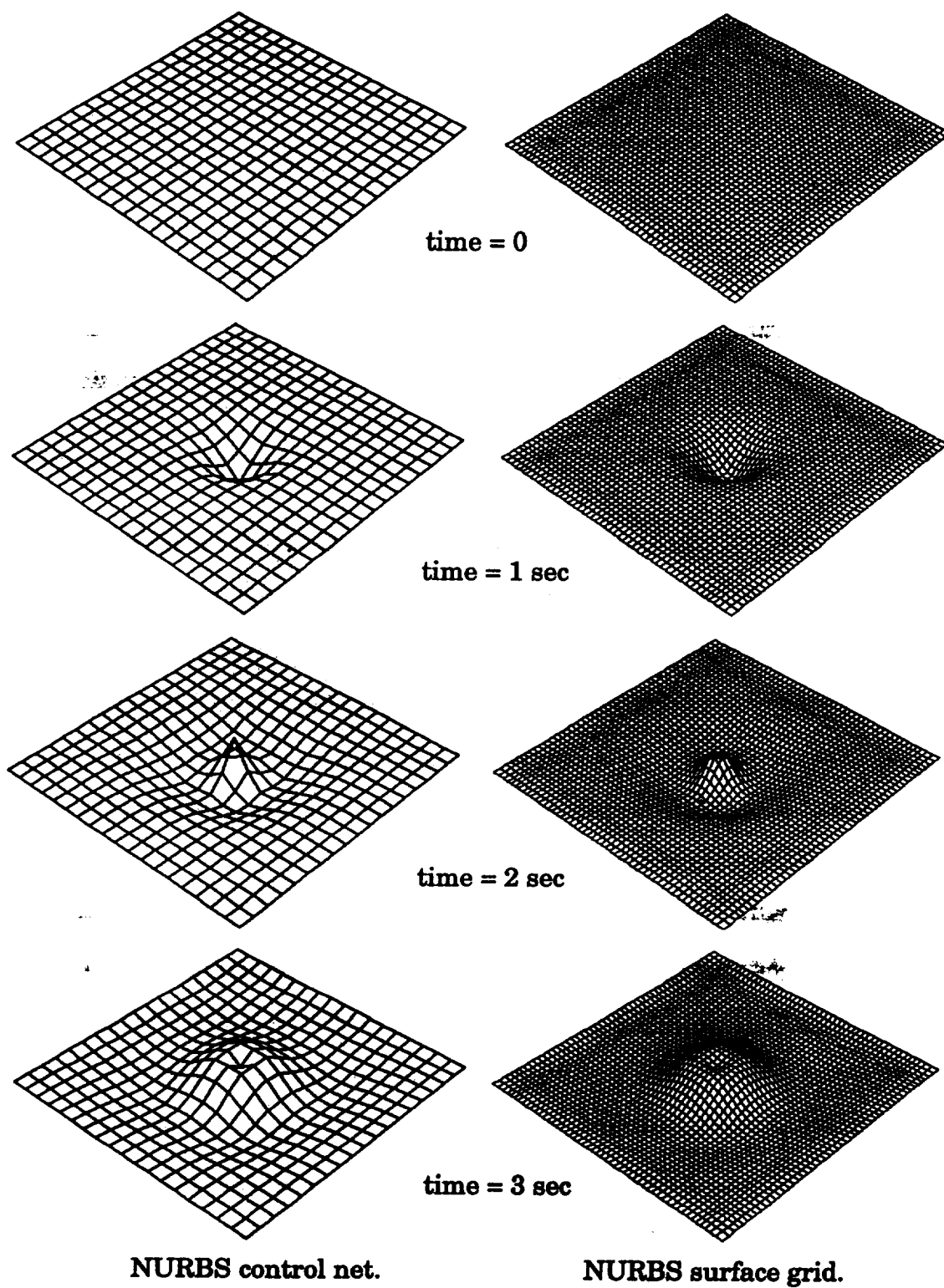
Figure 5.14 NURBS surface simulates the wave propagation.

than 90^0, three control points with proper *weights* will be enough to model the arc. The differences between these three entities are the determination of the *weights* and the values of the radii (or semi–major and semi–minor for the ellipse). Utilizing these three NURBS (circular arc, elliptic arc and superelliptic arc) representations, one is able to model a deforming grid — from a circular arc to elliptic arc (by changing the radii) and then to a superelliptic arc (by changing the *weights*). This example is demonstrated by a 3D transition duct. This 3D transition duct was designed to connect a typical circular engine exhaust to a high aspect ratio rectangular supersonic nozzle. The corresponding CFS simulation and the experimental results were first presented by the scientists in NASA Langley Research Center to investigate the effect of internal transition duct length on nozzle performance [Ref 41,48]. Many of the researches were concentrated on the CFS solution, however, the grids of this transition duct can be modelled easily by NURBS with a very concise control net (or control volume). Since the transitional cross sections were represented by the superellipse, the algorithm described in chapter two can be used to model each cross section with NURBS representation. Utilizing the "ruled volume" algorithm, one can generate the associated NURBS control volume for this circular–to–rectangular duct. Figure 5.15 shows the 3D grid as well as the associated NURBS control volume.
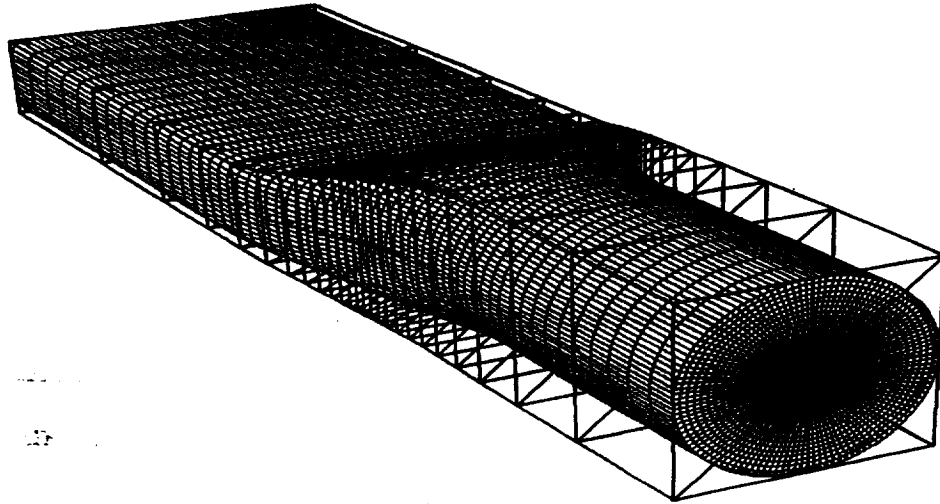
Figure 5.15 Circular–to–Rectangle nozzle (NURBS control volume & grids).

Based on the knowledge described in chapter two, one knows the fact that for a NURBS circular arc and elliptic arc with the same sector angle, the associated *weights* for these two NURBS representation are the same. The only difference is the radius of the circular arc changes to a pair of semi–major and semi–minor axes of the ellipse. Also, for a NURBS elliptic arc and a superelliptic arc with the same sector angle, the associated semi–major axes and semi–minor for these two NURBS representation are the same, however, the *weight* of the middle control point differs. Hence, one can apply these deforming relations to model this transition duct. The first procedure is to construct a NURBS cylinder with proper radius. A cylinder can be modelled by only 9 by 2 control points, however, in order to model the final circular–to–rectangular duct, more control points are needed in J direction so that it can catch the changing shape of the throat portion of the duct. Since only the radii change from the shape of circular cross sections to elliptic cross sections, the second step is then manipulating the associated radii of the control net at different cross sections to proper semi–major and semi–minor values without altering

the *weights*. During these processes, the shape of the cylinder will deform to the shape of circular–to–ellipse. The last step is then to increase the *weights* at the middle control point at each cross section of control net according to Table 2.1 for the superelliptic configurations. Figure 5.16 shows the deforming processes at different time steps for this configuration.

This grid generating process may be tedious and time consuming, which makes the simulation of deforming geometry problem difficult. In previous cases, the NURBS representation has been utilized for modeling the 1D and 2D wave equations. One may neglect the advantage of this NURBS modeling approach since the resolutions used for those two cases are not extremely high. However, for the volume case, the CPU time required for evaluating all the grid points is much larger than those of the curve and surface cases. Besides, the fact that the volume grids must be re–generated at each time step makes it a computational intensive problem. Utilizing the NURBS for this deforming geometry problem will make a substantial difference.

time = 0 sec

time = 0.2 sec

time = 0.28 sec

time = 0.41 sec
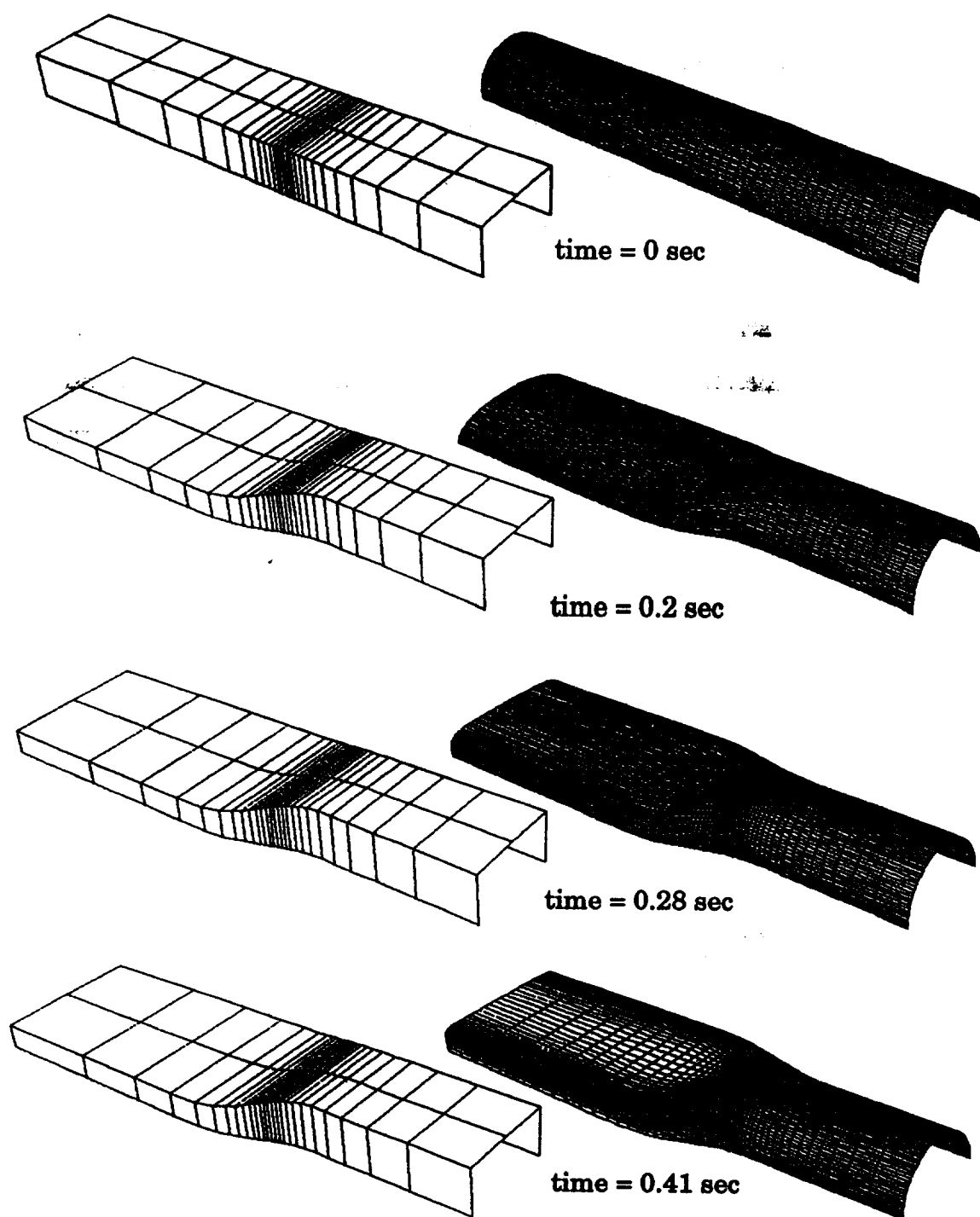
Figure 5.16 Dynamic grid generation for deforming duct geometry.

After demonstrating the capability of using NURBS to model the deforming geometry, an unsteady simulation using this approach is presented. This case simulates the flow passing through a 3D pipe which deforms at each time step. This 3D pipe is generated first by creating two NURBS cylinders with radii of r and 0 (a cylinder with radius 0 indicate a singular surface), the "ruled volume" algorithm is then applying to these two NURBS surfaces to create the final control volume. The size of control volume of this pipe is only $2 \times 2 \times 9$. However, in order to catch the deformation, more control points are needed in flow direction. The knot insertion algorithm is then used to achieve this goal. Figure 5.17 shows the control volume and the O type volume grid.
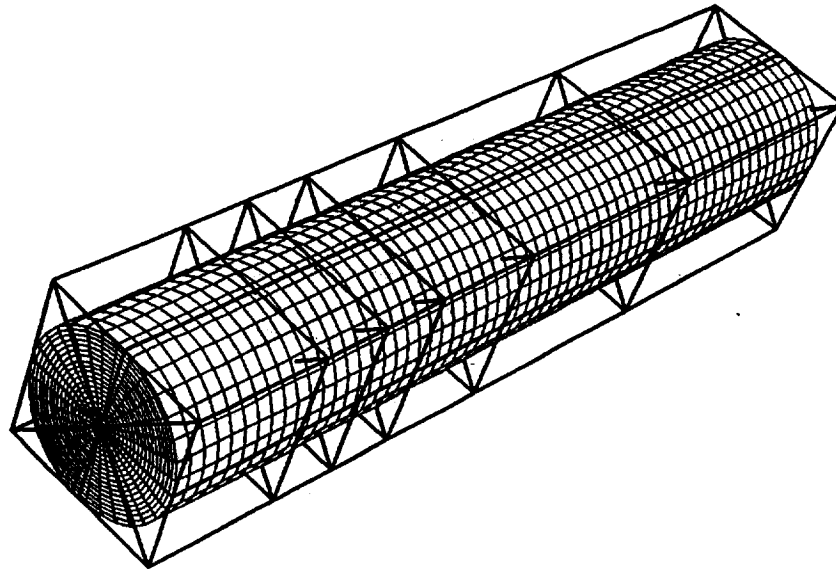
Figure 5.17 NURBS control volume for a deforming pipe.

Figures 5.18 and 5.19 show the deforming control volumes and the associated volume grids. The efficient NURBS evaluation algorithm (described in chapter 4) is utilized to generate the deforming grid at each time step. The unsteady solutions (courtesy Boyalakuntla [Ref 13]) obtained from *UBI* [Ref 2] flow solver are shown in Figure 5.20 ~ 5.22.
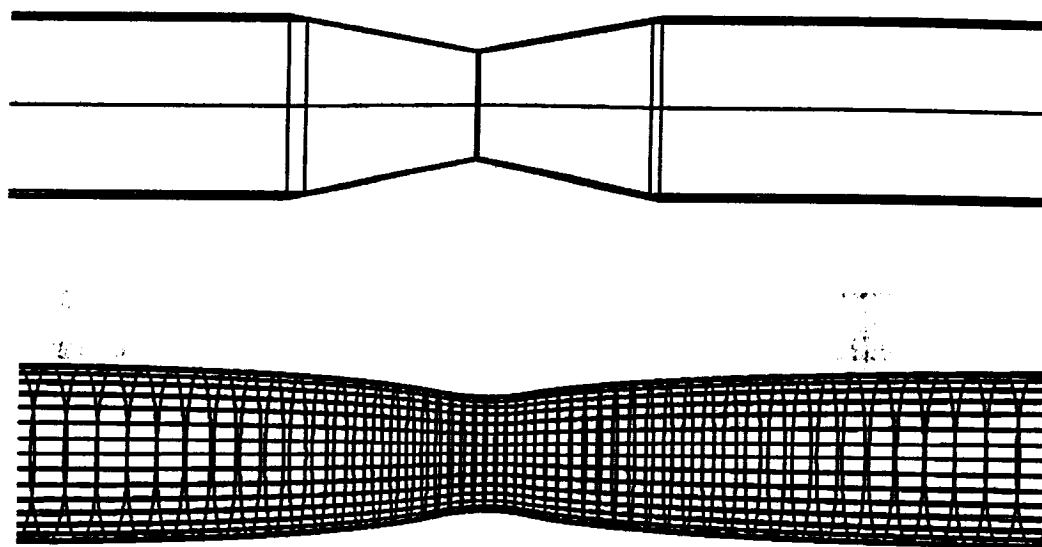
Figure 5.18 Control volume and grid of a deforming pipe (I).
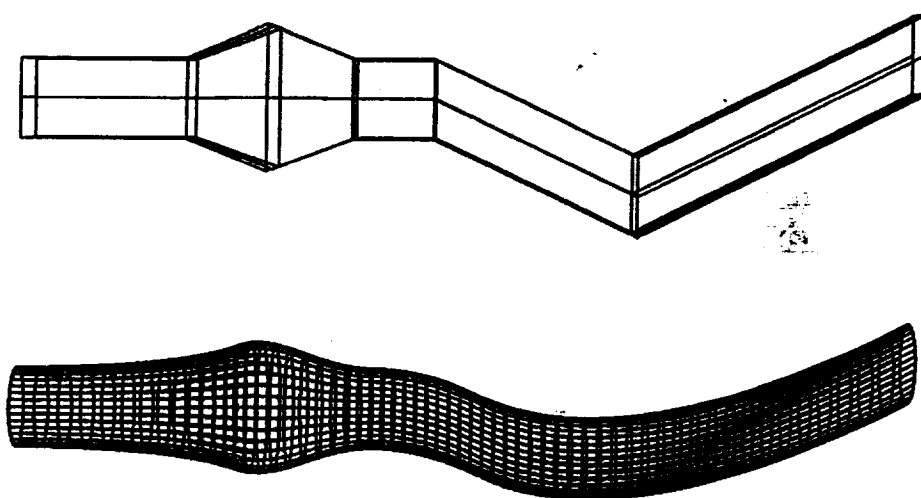
Figure 5.19 Control volume and grid of a deforming pipe (II).
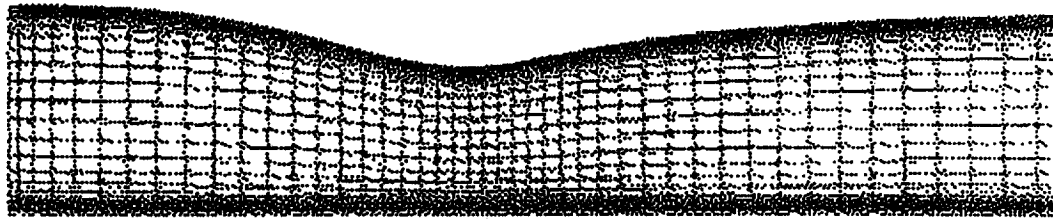
Figure 5.20 Grid and solution of a deforming pipe (I).
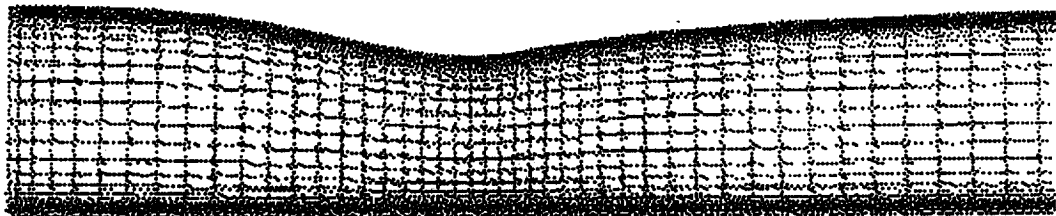
Figure 5.21 Grid and solution of a deforming pipe (II).
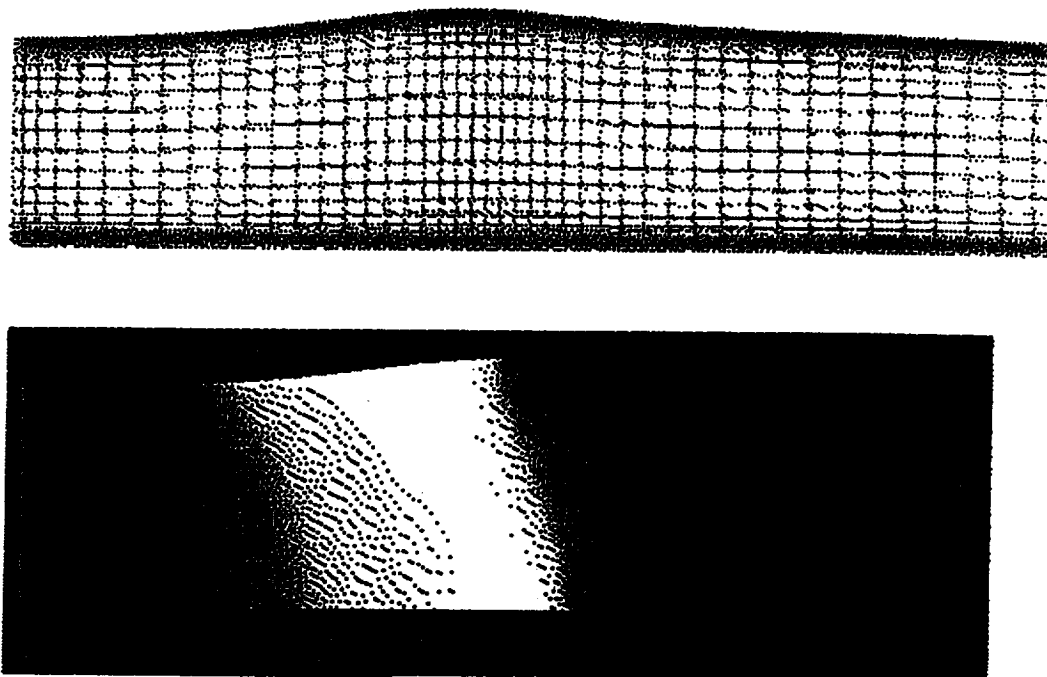
Figure 5.22 Grid and solution of a deforming pipe (III).

CHAPTER VI

CAGI: COMPUTER AIDED GRID INTERFACE

As is discussed in previous chapters, NURBS representation has many important properties which are useful in many research fields such as computer graphics, CAD/CAM, and the grid generation. It has also become a standard for geometric modelling in the industry. The transforming procedures and generation algorithms described in previous chapters show the versatility of using NURBS. The issue of geometry communication between the different packages is supported by the use of the standard IGES data format. In order to integrate all the NURBS properties and enhance the portability of geometry/grid, a computer software is developed to provide the efficient tools for CFS analysis. The structure and the current status of this software, *CAGI* — *Computer Aided Grid Interface*, is briefly introduced in this chapter.

CAGI Overview

CAGI is sponsored under the grant from NASA Marshall Space Flight Center. The motivation for developing this software is to try to build a bridge between the CAD/CAM system and grid generation codes. This code is aimed at developing a software system which integrates CAD/CAM geometric system output and IGES files for widely utilized grid generation systems and expecting it to allow for fast, efficient and economical responses to the pre–processing required in CFD applications.

144

The development of *CAGI* is carried out in a modular fashion by using the FORMS library [Ref 39] for designing the friendly Graphical User Interface (GUI). The main modules include:(i) Geometry transformer— developing a process of transferring widely utilized geometry entities defined in the IGES to NURBS format; (ii) Geometry manipulation – including NURBS geometric toolkits for easy manipulation of those sculptured geometries. The techniques of the "obstacles of using NURBS and the overcoming strategies" described in chapter 4 are also included to redistribute candidate geometric entities (curve, surface and volume) with desired distributions; (iii) Geometry generation – providing easy inputs for generating the NURBS sculpted geometry (curve, surface and volume). The overall objective of this work can be illustrated as Figure 6.1.
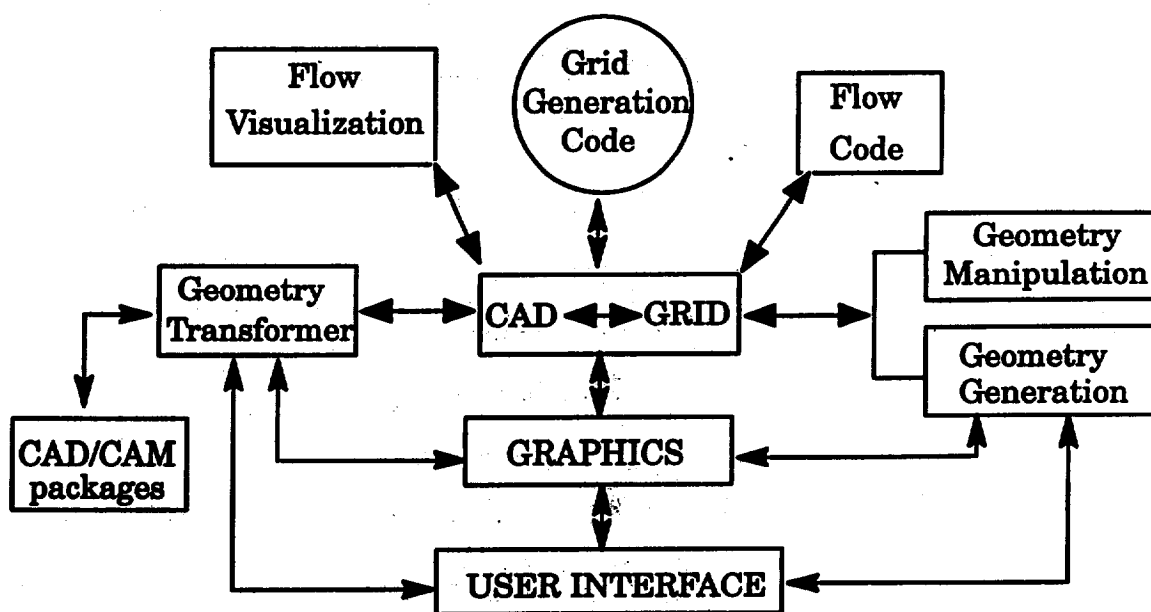
Figure 6.1 The modules of *CAGI* and their links.

The modules of "Flow Visualization" and "Flow Code" are not fully implemented in current *CAGI*.

Each module implemented in *CAGI* may be performed independently for different tasks. All of the modules are linked together with a common graphical user interface (GUI). This intuitive graphical user interface provides a user friendly environment for easy and efficient manipulation and generation of the desired sculptured geometry. The graphic visualization is developed utilizing the GL (Graphic Library) available on the SGI machines. Figures 6.3 and 6.4 show the graphical user interface with the entire screen layout of *CAGI*.

From Figures 6.3 and 6.4, one can realize that *CAGI* is constructed by creating a main panel window, an image display window, and several sub–panels. The main panel is the one which holds the commonly used function buttons or those buttons which can access the other sub–panels. The graphical image display window located in the middle of the main panel is the window that allows the user to interactively view the grids and the sculptured geometry. A message panel, located below the lower–left corner of the image display window, is designed to report any error message or provide some useful messages for the users. Next to this message panel are those function buttons which control the transformation. The user can adjust the proper sensitivity and utilize either those buttons or the mouse to control the translating, rotating and scaling of the displayed objects. Some other commonly used functions, such as refreshing the screen or changing the background color of the display window, are provided for the easy access on the global panel.

Transformer Module

As is discussed in the previous chapter, grid generation system and CAD/CAM design tools often utilized different formats which makes the transferrial of information between systems difficult and tedious. The Geometry

Transformer module designed in *CAGI* is developed to facilitate such communication. The theory behind this transformation is to use the NURBS as the interface. This is expressed in Figure 6.2.

| Discrete grid points | ←→ | (NURBS) | ←→ | IGES file |
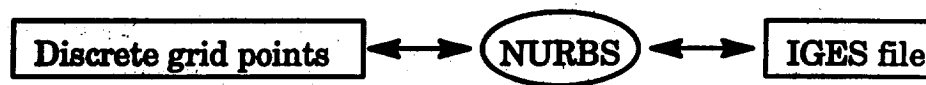
Figure 6.2 NURBS – interface between grid system and CAD/CAM.

Any standard IGES file (or NASA–IGES file and NINO) from a CAD/ CAM system can be transformed into NURBS and then output the desired geometric entities to discrete grid data set, such as the *PLOT3D* format, for the utilization in grid generation tools. Or from the other direction, any geometry defined by those geometry generation functions in *CAGI* can be output to a standard IGES for the use of any CAD/CAM design tools. The algorithms for transforming the various geometric entities into the NURBS representation are presented in chapters 2, 3 and 5. Table 6.1 lists those IGES geometric entities which can be transformed to NURBS definition in *CAGI* as well as those defined in NASA–IGES and NINO.

After an IGES file is read in, the geometric entities will be shown on the image display window, and the associated "name list" of each entities will be listed on the "Entity Name List" browser located on the left side of the main panel as shown in Figure 6.5. If the name list is highlighted, then the associated entity is defined as "active" and will be displayed in graphic window, while those without highlighting are defined as "inactive" and will be hided from the graphic window. The user can interactively turn on/off of certain entity by selecting the corresponding name list in this browser. However, no matter the entities are highlighted or not, they are all in the *CAGI's* database unless the user "delete" them by selecting the proper function.
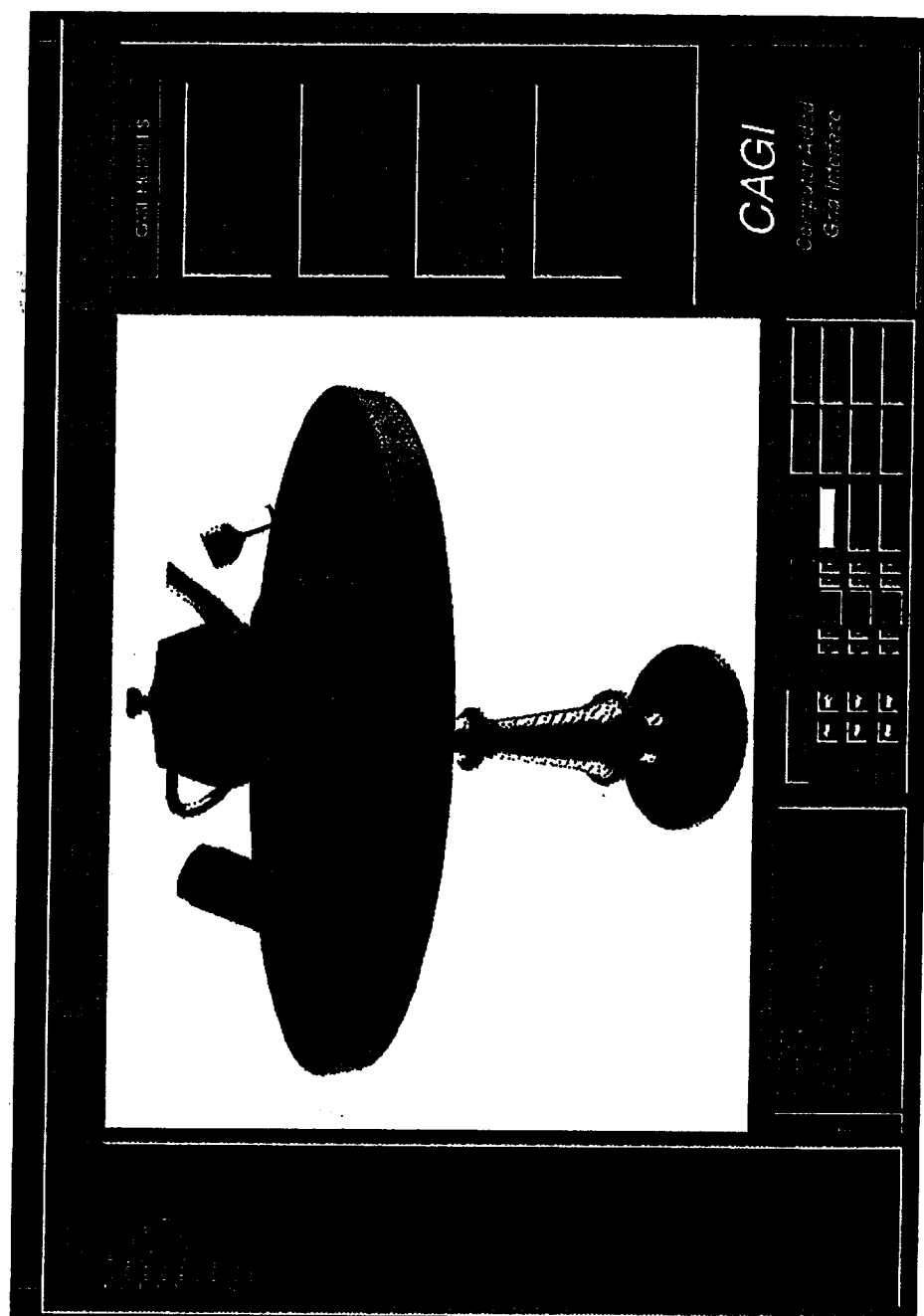
Figure 6.3 *CAGI* screen layout (1).

Figure 6.3 *CAGI* screen layout (1).
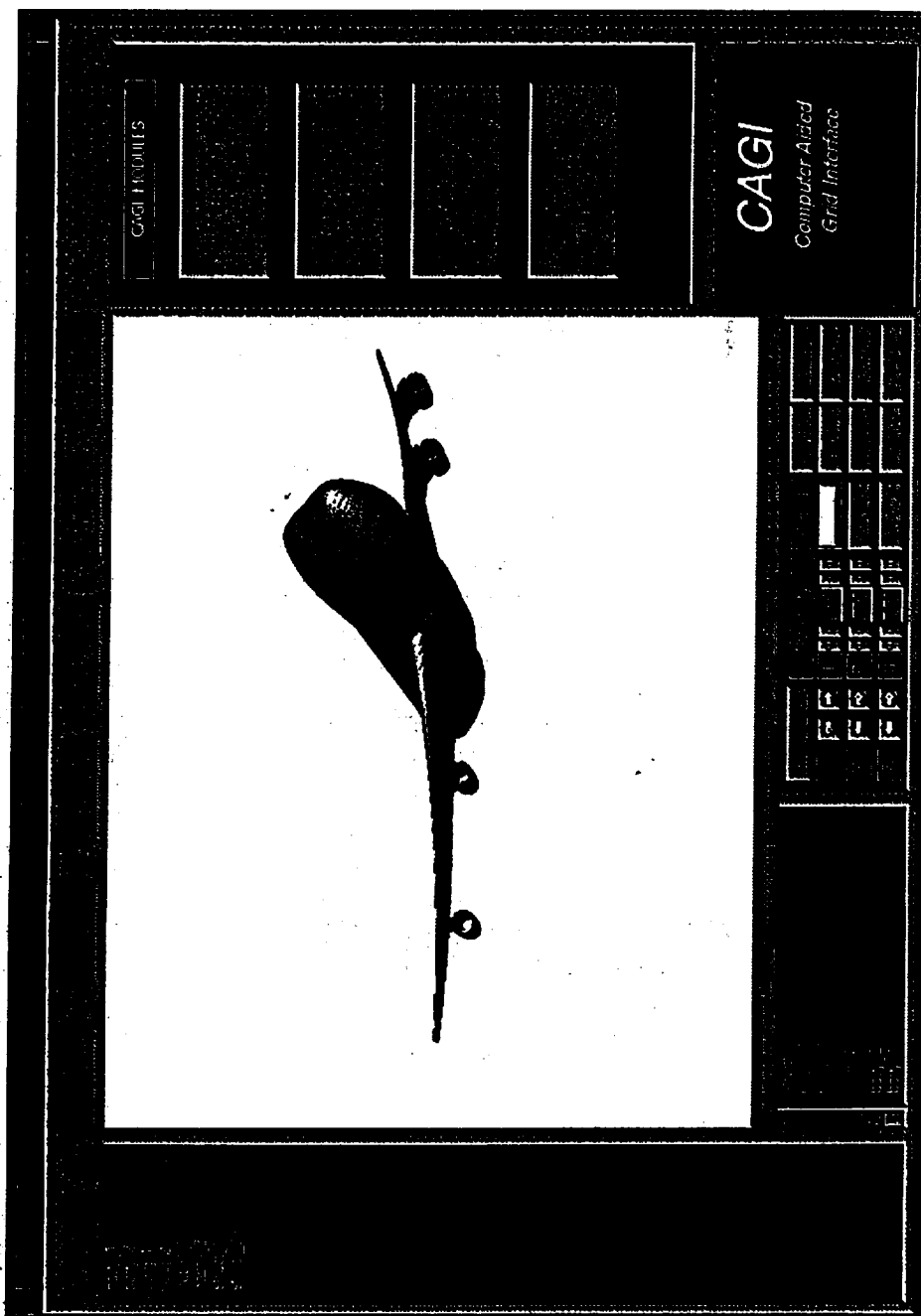
Figure 6.3 CAGI screen layout (2).

Figure 6.4 *CAGI* screen layout (2).
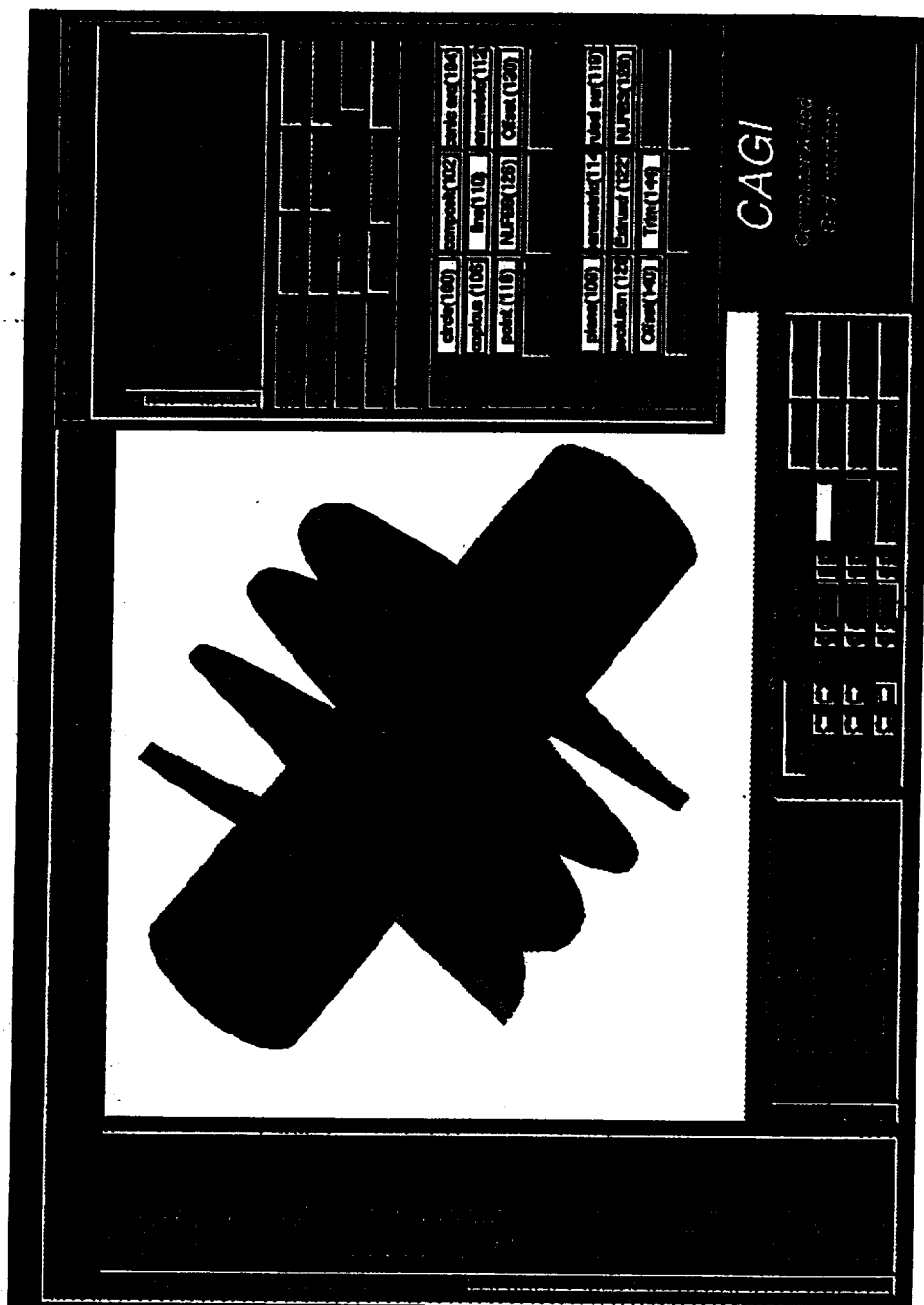
Figure 6.5 Illustration of IGES Transformer Module in CAGI.

Table 6.1 List of IGES Entities Supported by Different Packages.

Entity Number	Entity Name	NASA IGES	NINO	*CAGI*
0	Null Entity	✓	✓	✓
100	Circular Arc Entity	✓		✓
102	Composite Curve Entity	✓		✓
104	Conic Arc Entity	✓		✓
106	Copious Data Entity	✓		✓
108	Plane Entity			
110	Line Entity	✓		✓
112	Parametric Spline Curve Entity			✓
114	Parametric Spline Surface Entity			✓
116	Point Entity	✓		✓
118	Ruled Surface Entity			✓
120	Surface of Revolution Entity			✓
122	Tabulated Cylinder Entity		✓	✓
124	Transformation Matrix Entity	✓	✓	✓
126	NURBS Curve Entity	✓	✓	✓
128	NURBS Surface Entity	✓	✓	✓
130	Offset Curve Entity			✓
140	Offset Surface Entity			✓
141	Boundary Entity	✓	✓	✓
142	Curve on a Parametric Surface	✓	✓	✓
143	Bounded Surface Entity	✓	✓	✓
144	Trimmed Surface Entity			✓

The fields without "✓" under NASA–IGES and NINO don't mean these translators can not process those entities, it only indicates those entities must be transformed first to any entities allowed in NASA–IGES and NINO format. And the "Bounded Surface" and "Trimmed Surface" must be processed interactively in *CAGI*.

In the process of designing the sculptured geometry in the CAD/CAM systems, many geometric entities which are not necessary for the CFS analysis may be created. For example, a surface may be formed by revolving a boundary curve which was defined by set of discrete points, but the final IGES file created by the CAD may contain all the points and curves which are not needed. Reading all the entities and transforming them to NURBS definition requires a lot of CPU time and memory for those un-interested entities. As well displaying all of the entities on the graphic windows not only clogs the entire screen but also makes it hard for any manipulations. *CAGI* provides a solution to this problem by allowing the user to filter the entities before they are processed. For example, turn off entity 106 and 110 will make *CAGI* to skip processing the points and lines contained in an IGES file. And the other alternative to avoid the hurdle is to read all entities and store them in database, but use the "group function" to turn on/off or delete groups of entities. For example, the user can turn on/off all the points, curves or surfaces or certain specific entity type by selecting the proper buttons. A status report regrading the IGES entities is provided after this procedure.

Geometry Manipulation Module

After transforming an IGES file, one may select this module to modify the geometry by utilizing the NURBS properties. The NURBS information such as the degree, number of control points of the selected curve or surface will be shown in the proper position of this panel. The user can change the control polygon coordinates or the associated *weights* and visualize the change of the geometry interactively. Nevertheless, the most useful features for the NURBS representation in the applications of the numerical grid generation are the changing of resolutions and the distribution functions without distort-

ing the original geometric definition [Ref 88,90,91]. When the user select this module, a distribution panel also pops up at the right–lower corner of the main interface. This distribution panel provides eight different packing functions to define the spacing of the NURBS curves, surfaces and volumes. However, as is aforementioned, the largest obstacle which inhibits the engineer to utilizing the NURBS definitions is the difficulty in controlling the spacing of the NURBS boundary or surface. This is because the NURBS is represented in the parametric form, the distribution function defined in parametric space may not always be reflected in the physical space. In other words, an even spacing in the parametric space may not result in an even distribution in physical space (more detail information about this problem is discussed in chapter 4). To solve this problem, *CAGI* provides options for the user to choose whether the distribution is defined in parametric or physical space. The sharp corners (discontinuous points) on the curves or surfaces can also be maintained during the redistribution procedure. Moreover, the NURBS tool kits, such as knot insertion and degree elevation, data reduction are implemented. When reading the IGES file, the geometric entities may have different orientations which make them difficult for CFS analysis. For example, two adjacent surfaces may have complete different I and J directions. For this problem, *CAGI* also provides the functions which allow the user to "reverse" the directions, "swamp" the orientations of any selected NURBS entities. Figure 6.6 shows the functions implemented in this module.

Geometry Generation Module

CAGI not only imports the geometry defined in the CAD/CAM system, it also has the capability to create geometry with NURBS definition for 3D curves, surfaces and volumes. Since the grid points / lines may not necessarily

pass through the NURBS control polygons (or the control nets), it has been a difficulty for grid generation software to model the geometry utilizing the NURBS information. For example, most of the engineers who don't have a solid NURBS background don't know how to define the NURBS control net, the associated *weights* and the knot vectors to model a sphere (or other geometry). However, in *CAGI*'s generation module, this problem has been overcome. The user doesn't even have to know the NURBS definition to generate the boundaries and surfaces (volumes). The corresponding NURBS definitions (control points, *weights*, *orders* and knot vectors) will be automatically created by the user's simple input. For example, to build a NURBS circular arc, the user only need to specify the center, the radius and the desired angles, *CAGI* automatically transforms these specified information to the NURBS representation. Same for the generation of the surface and volume. The functions implemented in this module include the generations of point, curve, surface and volume. For NURBS generation functions, they are listed as follows.

NURBS Curve Generation Functions:

The functions implemented in this module include generating of the NURBS "Straight line", "Circular arc", "Conic arc", "Reading a NURBS curve", "Extract a NURBS curve from a NURBS surface", "Interpolate data set to a NURBS curve", "Translate / Scale / Rotate of a existed NURBS curve", "Rotate a point with any arbitrary axis to create a NURBS arc", "Split a NURBS curve to two NURBS segments", "Join two NURBS curves to one NURBS curve", "Create a NURBS curve by trimming a existed NURBS curve", "Creating a NURBS curve which lay on top of a NURBS surface", "Duplicate a NURBS curve with respect to any arbitrary axis", "Reflect a NURBS curve with respect to any arbitrary plane".

NURBS Surface Generation Functions:

The functions implemented in this module include generating of the "NURBS ruled surface", "NURBS surface of revolution", "NURBS Tabulated cylinder (Extruded surface)", "Reading a NURBS surface", "Translate / Scale / Rotate of a existed NURBS surface", "Transfinite Interpolation of a NURBS surface", "Duplicate a NURBS surface with respect to any arbitrary axis", "Interpolate data set to form a NURBS surface", "Split a NURBS surface to several NURBS sub patches", "Join two NURBS surfaces to one single NURBS surface", "Create a NURBS surface by trimming a existed NURBS surface", "Reflect a NURBS surface with respect to any arbitrary plane" and "Offset NURBS surface".

NURBS Volume Generation Functions:

The functions implemented in this module include generating of the "NURBS ruled volume", "NURBS volume of revolution", "NURBS Tabulated volume", "Reading a NURBS volume", "Translate / Scale / Rotate of a existed NURBS volume", "Transfinite Interpolation of a NURBS volume", "Duplicate a NURBS volume with respect to any arbitrary axis", "Split a NURBS volume to several NURBS sub volumes", "Join two NURBS volumes to one single NURBS volume", "Create a NURBS volume by trimming a existed NURBS volume" and "Reflect a NURBS surface with respect to any arbitrary plane".

Figure 6.7 shows several NURBS surfaces created in this generation module.
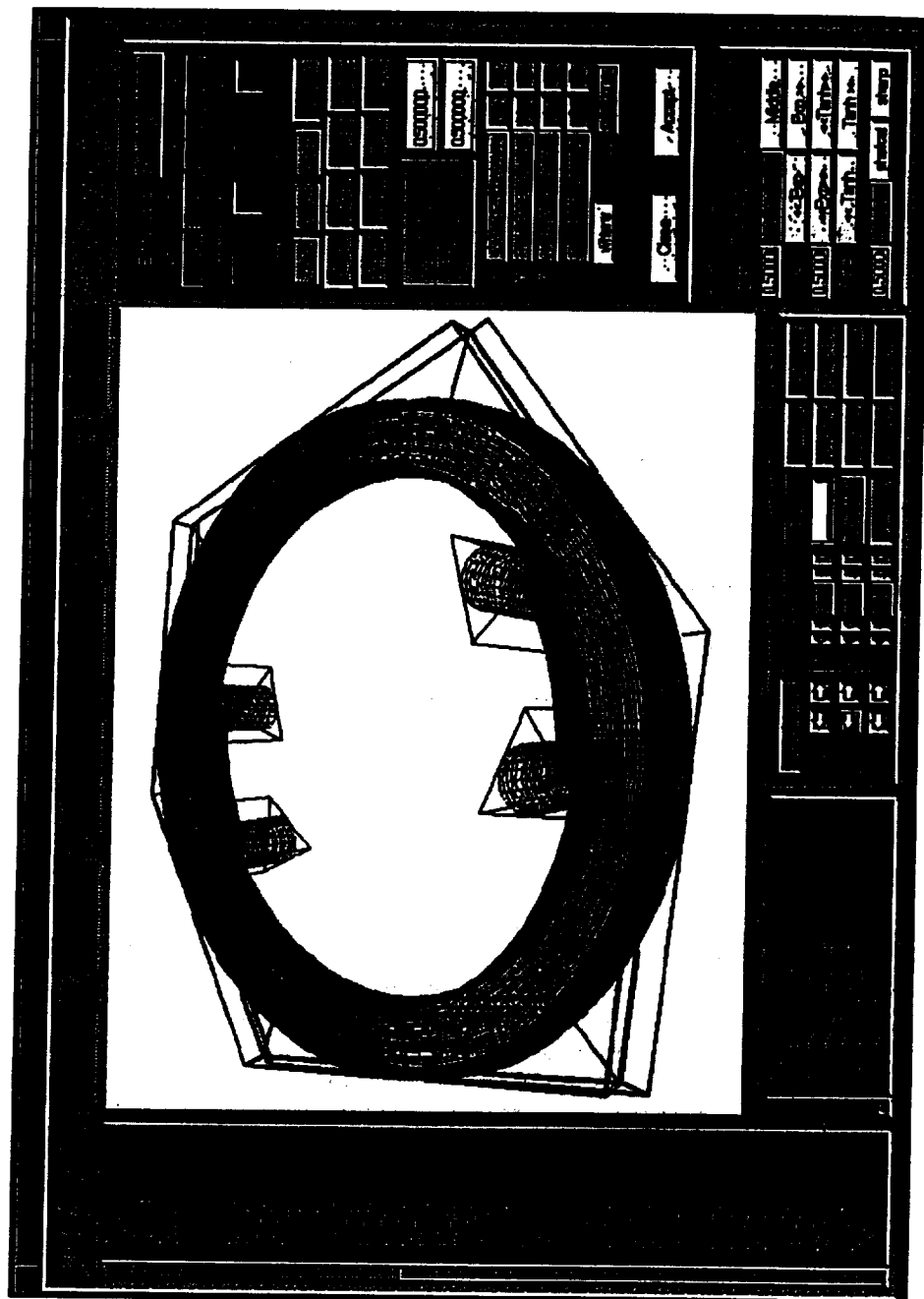
Figure 6.6 Illustration of NURBS Manipulation Module in CAGI.
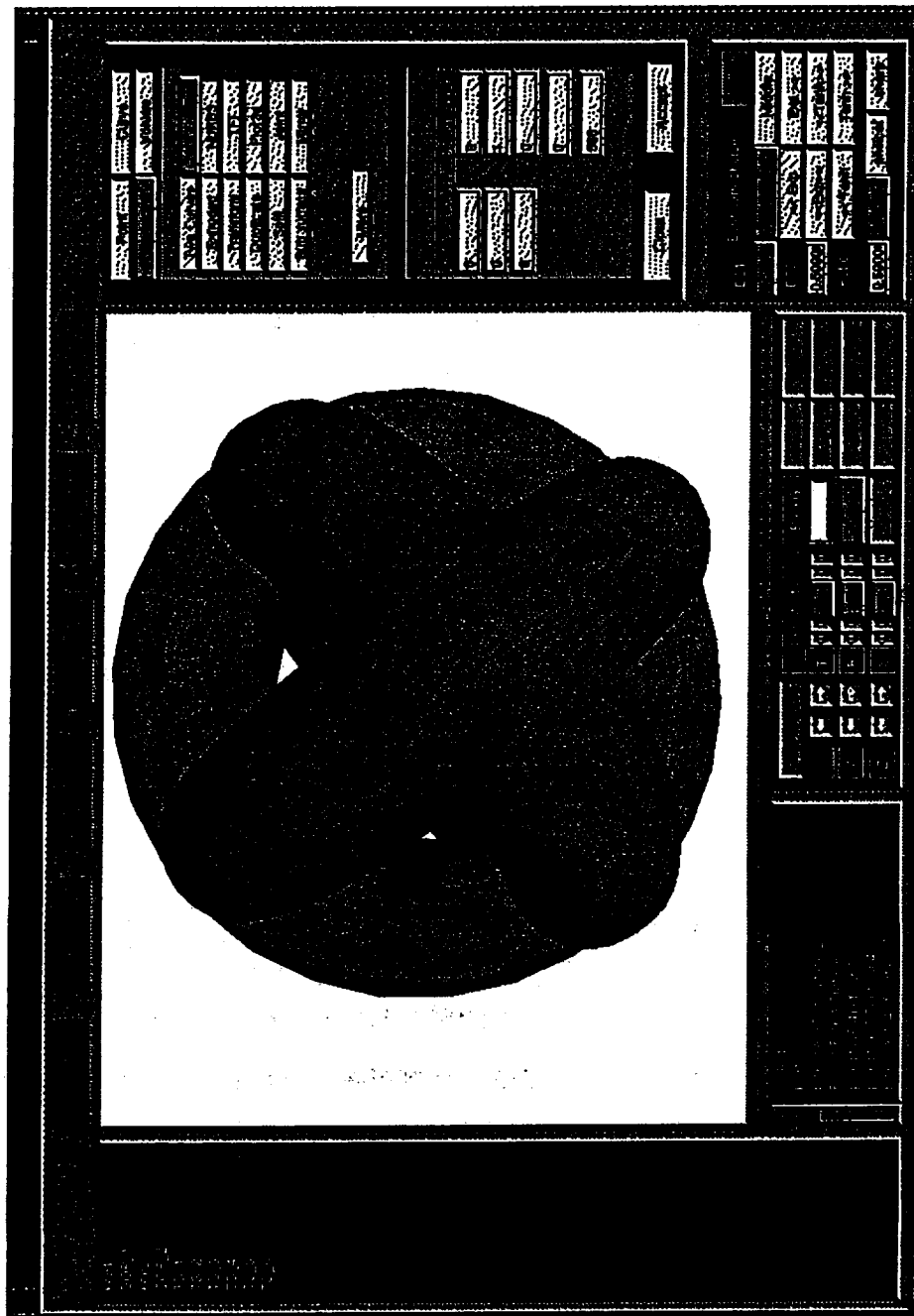
Figure 6.7 Illustration of NURBS Generation Module in CAGI.

Figure 6.7 Illustration of NURBS Generation Module in CAGI.

CHAPTER VII

STRUCTURED GRID TOPOLOGY ON TRIMMED SURFACE

As mentioned earlier, CAD/CAM systems generally utilize the IGES format for geometry Input and Output. The trimmed parametric surface (entity 144) is the most frequently used entity in the IGES format. Unfortunately, this entity is also the most difficult entity for structured grid topology. This trimmed entity can not exist alone, it must be coupled with another entity — entity 142, which is the curve on a parametric surface entity. This entity is used as the trimmed curves to define the boundaries of the trimmed surface. Hence, to know how a trimmed surface is defined, entity 142 must be presented first.

Curve on a Parametric Surface Entity (Type 142)

This entity is defined by a parametric surface, called an "untrimmed" surface along with the parametric curve, and identifies the curve as the entity lying on top of the surface [Ref 35].

Let the domain D of the parametric surface be a rectangle in (u,v) space, then the surface $S(u,v)$ can be defined as equation (7.1).

$$S(u,v) = (x(u,v),\ y(u,v),\ z(u,v))$$
$$D = \{ (u,v) | u_1 \leq u \leq u_2,\ v_1 \leq v \leq v_2 \}$$

(7.1)

The domain of the parametric curve, denoted as $C(t)$, must lie in the range of the parametric domain of the surface D, and assume the parametric values which are used for evaluating $C(t)$ are represented as $B(t)$. Then $B(t)$ must

satisfy the equation (7.2) shown as follows.

$$B(t) = (u(t), v(t)) \quad a \le t \le b \quad a \in [u_1, u_2], \quad b \in [v_1, v_2] \qquad (7.2)$$

Then the parametric curve $C(t)$ which lies on top of the surface can be represented as equation (7.3)

$$\begin{aligned} C(t) = S(B(t)) = S(u(t), v(t)) = \\ (x(u(t), v(t)), \ y(u(t), v(t)), \ z(u(t), v(t))) \end{aligned} \qquad (7.3)$$

Hence, there are three geometric entities included in entity 142. They are the untrimmed surface, parametric curve $B(t)$, and the parametric curve $C(t)$. The curve $B(t)$ is a 2D plane curve and lies on the parametric domain of the untrimmed surface, while the curve $C(t)$ is a 3D curve lies on the untrimmed surface in physical space. These two curves are mutually related — $C(t)$ can be generated by the composite mapping of $S(B(t))$, while $B(t)$ can also be obtained by projecting curve $C(t)$ onto the untrimmed surface and finding the corresponding parametric values in the parametric domain. Because these two curves are both given in the IGES file, there may be conflicts if one like to specifies one curve differently from the other (say, obtaining the curve $B(t)$ from curve $C(t)$ vs. obtaining curve $C(t)$ from $B(t)$). This means projecting the curve $C(t)$ onto the untrimmed surface may not be able to obtain a curve exactly the same as $B(t)$, or on the other hand, evaluating the $B(t)$ on surface $S(u,v)$ may not be able to create the same curve as $C(t)$. Obviously, this will lead to the accuracy problem. Hence, in an IGES file, another flag is used to control which curve should be used to avoid this problem. This flag is denoted as "PREF" (preferred flag). If PREF is 0, IGES does not specify which curve to use, if PREF is 1, the $B(t)$ should be used, if PREF is 2, then curve $C(t)$ is pre-

ferred, if *PREF* is 3, then both *C(t)* and *B(t)* are equally preferred. Figure 7.1 illustrates the entity 142.
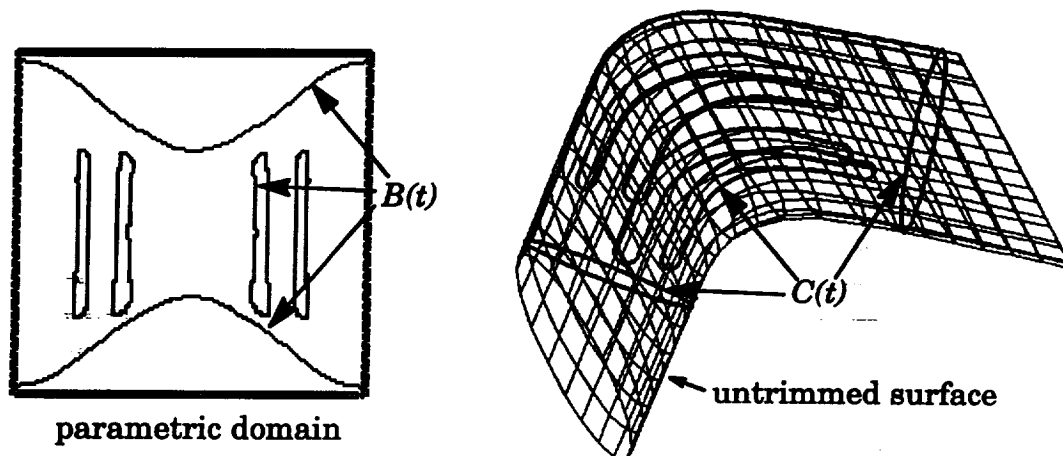
Figure 7.1 Illustration of IGES entity 142.

After knowing the format of entity 142, the trimmed surface can be defined. In the IGES format, the trimmed surface is defined by a surface, called the untrimmed surface, along with two types of boundaries — the outer boundary and inner boundary which are lying on the untrimmed surface. Both types of boundaries are defined with the format of entity 142. Hence, the untrimmed surface which the boundaries (entity 142) lie in must be identical to the one defined in the trimmed surface entity. If the number of inner boundaries is greater than or equal to zero, however, there is only one outer boundary defined. If there is no outer boundary curve specified, the boundaries of the untrimmed surface will be used as the default outer boundary (one closed curve), and if there are no inner boundaries and no outer boundary specified, then the trimmed surface will be equal to the untrimmed surface. Furthermore, all inner / outer boundaries must be simple closed curves and mutually disjoint. Each of the inner boundaries must lie in the interior of the outer boundary. Hence, the domain of the trimmed surface is defined as the com-

mon region of the interior of the outer boundary and the exterior of each of the inner boundaries and includes the boundary curves. Figure 7.2 illustrates the definition of this entity.

Trimmed surface region
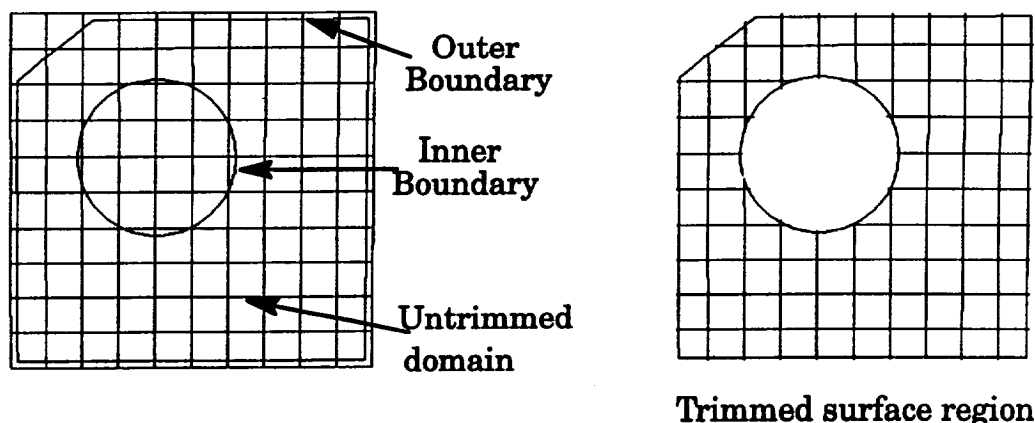
Figure 7.2 Illustration of IGES entity 144.

The four–sided structured patch requires the number of grid points on the opposite boundary be the same; it also requires that the total number of grid points be equal to the multiplication of the grid points of two adjacent boundaries. Apply these two criteria to the trimmed surface shown in Figure 7.2, one can immediately understand that the trimmed surface is not a real structured patch. Indeed, it is shown only for display. As matter of fact, it is plotted by the "scan lines" and the "filling polygons" algorithms [Ref 27]. Because all the inner and outer boundaries are simple closed curves, any "scan" line which parallels to I (or J) direction of the parametric domain will intersect all the boundaries with even number of points. Out of all these intersection points, those line segments starting with odd number will be plotted. For example, plotting the line segments connecting the points of (1, 2), (3, 4), ... (N–1, N). Same procedure should be applied to the J (or I) direction for another set of line segments. After this algorithm, the trimmed region (bounded by the

interior of outer boundary and the exterior of inner boundaries) can then be displayed, however, it can not be utilized for structured grid generation.

To grid the trimmed surface for a "structured" topology is difficult. The reasons for this are summarized as follows:

Complexity

As is discussed in previous section, the trimmed surface (entity 144) defined in the IGES file may contain many inner boundaries, also the boundaries could be any shape (convex or concave curve). For the structured grid, it requires 4-sided boundaries. However, in many of the trimmed surfaces, there are 3-sided, 5-sided, or even more than 6-sided boundaries. This phenomenon makes it very complicated for structured grid generation. It is almost impossible for the "automatic" structured grid generation for a complicated trimmed surface. Figure 7.1 demonstrates the complexity of the structure grid generation, in which the trimmed surface contains four inner boundaries and the outer boundary is not convex.

Inconsistency

The trimmed surface shown in Figure 7.2 is a simple geometry. It contains only one inner boundary and one outer boundary. However, one can not create the "ruled" surface for the trimmed region by just connecting the two boundaries. The reason of that is the two boundaries have different directions and the starting points of the two curves are far away from each other. The directions and the location of the starting points of the boundaries create the inconsistency problem which increses the difficulty for griding the trimmed surface.

Accuracy

Another difficulty for griding the trimmed surface is related to the accuracy problem. Taking Figure 7.2 as an example, the inner boundary should be a closed curve, however, the starting point and the ending point of this curve are not exactly identical. The gap between these two points may come from the tolerance defined in different design systems. This accuracy problem also happens to the gap between the common edges of two adjacent trimmed surfaces. Also, all the trimming boundaries are defined with the format of entity 142 (curve on parametric surface), if improper parametric values are used, the sharp corners (discontinuous points) on the boundaries may be lost, and this will lead to an inaccurate geometric definition. Because the trimmed surface was designed in CAD/CAM systems, the accuracy problem already existed when the geometry was imported to grid generation system. This increases the difficulty for generating grids for the trimmed surfaces.

Grid Topology Requirement

As it has been mentioned it is very difficult to design an automatic griding package for a structured trimmed surface. Even though the simple trimmed surface may be handled automatically, the returned structured grid may not be useful for all the solution algorithms. This is because different solution algorithms may require different grid topologies – such as H type to avoid the singularity point. Due to the preferred grid topologies required, the difficulty for griding the trimmed surface increased.

All these difficulties indicate the design of an automatic structured grid generator for the trimmed surface (entity 144) would be impracticable. In order to overcome these obstacles and also let the engineer determine the desired grid topology, CAGI currently designs several panels for the user to

create structured grids interactively. The functions included in these panels are described as follows.

Resolution Control

The first panel is developed for the purpose of visualizing the trimmed entity. As is discussed in previous section, the untrimmed surface and the trimming boundaries are presented as NURBS format. Hence, the resolutions will control how the geometry be displayed on computer screen. If the resolutions of the trimming entities are too low, the shape of the geometry may not be truly displayed. On the other hand, if the default resolutions are set too large, the program will take much CPU time and more memory to evaluate all entities. Since the engineer is only interested in the "trimmed" region, setting large resolutions for untrimmed surfaces and the trimming boundaries will simply waste the computer resources. Therefore, the motivation for designing this panel is to provide easy tools to increase the resolutions for a better understanding of the geometry, or decrease the resolutions to reduce the memory load after the trimmed region is completed. Figure 7.3 shows the layout of this panel.

Figure 7.3 Resolution control panel.

Breaking Point Define

In order to generate the structured grid on the trimmed surfaces, it is necessary to do the domain decomposition, and the first step of the decomposition is defining the breaking points on the trimming boundaries. The functions implemented in this panel allow the user move the slider bar and interactively see a breaking point moving on the trimming boundary both in the parametric and physical space. The parametric value and the coordinate of the breaking point will be updated whenever the user moves the slider bar. One can press the "*Add*" (or "*Remove*") button to insert (or delete) the breaking points from the database. Moving the slider bar may fail to obtain the proper parametric values to catch the sharp corners on the trimming boundary. Hence, a function is implemented to allow the user obtain all possible sharp corners by finding the proper parametric values. These functions are illustrated in Figure 7.4.
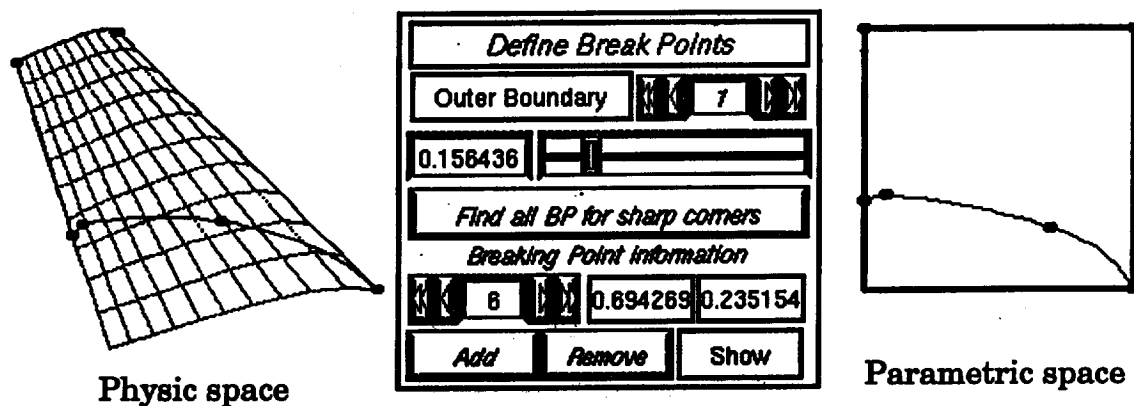
Figure 7.4 Break point define panel.

Edge Define

After the breaking points are defined, the user is allowed to use any two breaking points to define an edge. The edge could be a straight line in the

parametric domain or a curve along the trimming boundary. While defining the edges, the user not only can decide the resolutions but also can chose the desired distribution by selecting various packing functions. The distribution is referred to the physical space which means the user's desired distribution is reflected on the trimmed surface region. The commonly used functions, such as "Reverse the edge direction", "Combine two edges" are also implemented for a easy manipulation. Figure 7.5 shows these functions and the corresponding edges in the parametric space as well as the physical space.

Physical space
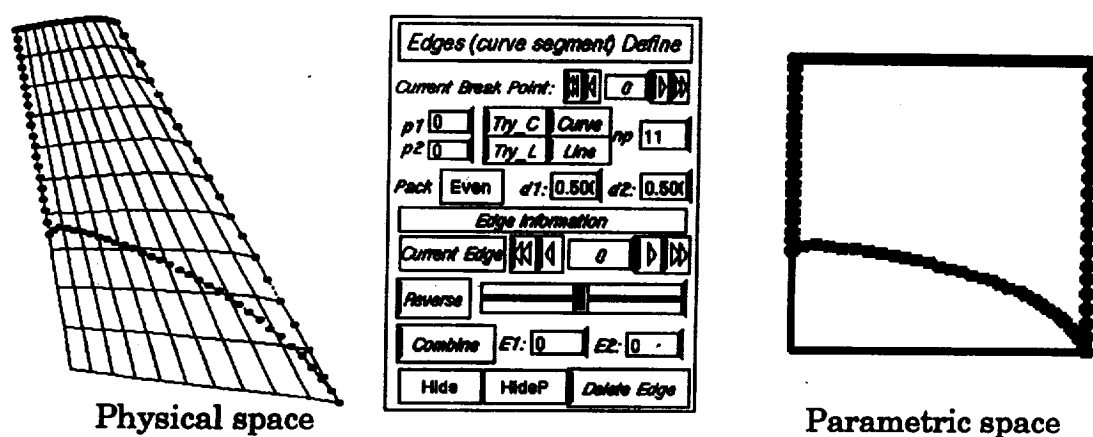
Parametric space

Figure 7.5 Edge define panel.

Structured Patch Define

After the edges are proper defined, the user can select four edges which form a closed region and then apply the *TFI* algorithm to these four edges to create a structured patch in parametric space. One can obtain one piece of trimmed surface in the physical space by utilizing the parametric patch as the parametric values and evaluating them with the untrimmed surface. Because the *TFI* is applied to the edges on the parametric space, the resulting patch in parametric space may not always result in a smooth surface in physical space.

Hence, the elliptic function is implemented so that the user can improve the quality of the final trimmed surface. These functions and example are all illustrated in Figure 7.6.

Final trim surface.

TFI Patch Define

create	Edge ID:		0	
smooth	E1	E2	E3	E4
delete	1	2	3	4
Hide	color	patch id	0	
Output to file	Read in Info	data.out		

TFI parametric patch.
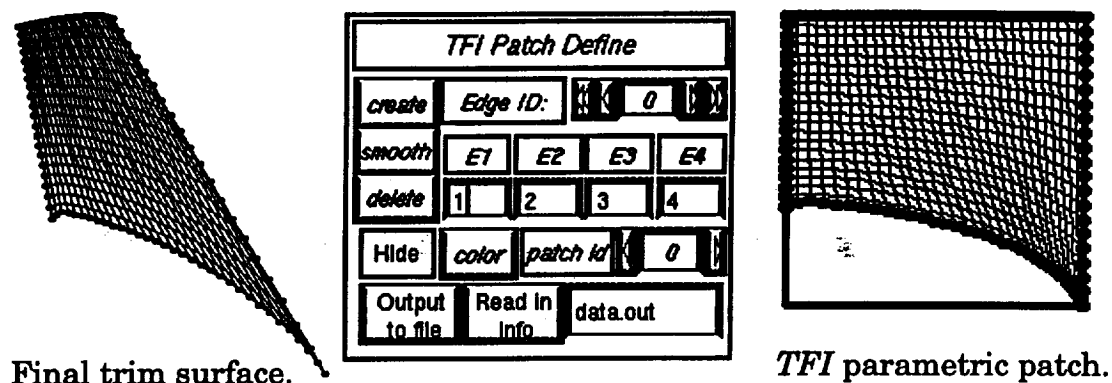
Figure 7.6 TFI patch define panel.

After the user generates the pieces of trimmed surfaces, the output functions designed in *CAGI* can be selected for outputing the surfaces to different data formats (for example, the PLOT3D format), the grid generation package, such as the *GENIE++* [Ref 61], can use there surfaces for the volume grid generation.

These interactive procedures allow the user to generate the structured grids with desired topology on the trimmed surfaces. The algorithm which can catch the discontinuous points (described in chapter four) has been installed in "Break point define" panel. It allows the user obtain the proper parametric values for the sharp corner in physical surface, hence, the proper break points can be defined. The re–parameterization algorithm (described in chapter four) which can generate the distribution accurately on physical space has been installed in the "Edge define panel". It allows the user to specify the desired point distribution on the edges in trimmed surfaces. The smooth algorithm based on the elliptic function has been implemented in the *"TFI* patch

panel" for a satisfactory structured grid. Another structured trimmed surface example obtained from these procedures are shown as follows.

un–trimmed surface   parametric domain   trimmed surface
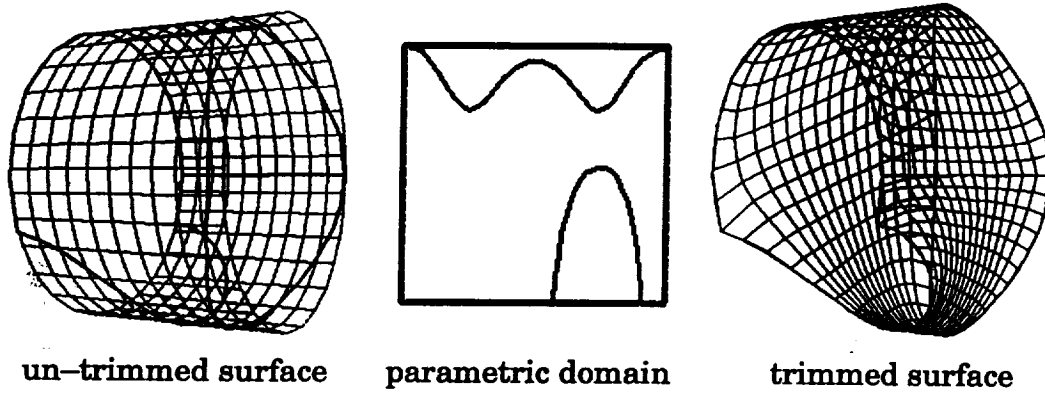
Figure 7.7 Trimmed surface example.

CHAPTER VIII

SUMMARY AND CONCLUSIONS

The geometry modeling techniques used in Computer Aided Geometry Design have been extended and applied to numerical grid generation for complex real world applications. The general transformation algorithms which precisely convert the Non–NURBS entities to NURBS representations have been developed. These algorithms can be utilized to bridge the gap between the grid generation and the CAD/CAM systems. Also, the two–way linkage between the CAD/CAM and grid systems has been achieved through the IGES format utilizing entities 126 (NURBS curve) and 128 (surface). The formulation of NURBS has been extended from curves, surfaces to full 3D NURBS control volumes to model the CFS configurations. The development of the re-parameterization schemes and their influence on the curve / surface / volume grid distributions is demonstrated by computational examples. The applications of these re–parameterization techniques to precise grid distribution control with accurate geometry fidelity have been demonstrated. An efficient NURBS evaluation routine has been developed to facilitate the entire grid generation process. In addition, the applications of NURBS to dynamic grid generation presented in this study have proven the versatility of NURBS in the CFS simulation processes.

The algorithms developed in this study have been utilized for various grid generation related research areas. For example, the interpolation routines developed under this research have been applied to Yang's adaptation

[Ref 83,84,85] as well as Shih's Turbomachinery grid generator *"TIGER"* [Ref 52,56]. The NURBS transformation and evaluation routines also have been applied to Boyalakuntla's unsteady simulation for temporally deforming geometries [Ref 13]. These algorithms and the associated CFS applications have been published in the Journal of *"Computer-Aided Design"* and *"Applied Mathematics and Computations"* [Ref 51,94].

The NURBS generation and manipulation algorithms have been incorporated as modules in the CAGI system. The motivation for developing CAGI has been to follow the IGES standard, and incorporate the NURBS representations for the two way communications between the CAD/CAM and grid systems. CAGI provides the capability of processing the IGES files and converts various IGES entities to NURBS representations. These NURBS representations are then evaluated, thus preparing the boundaries and surfaces for grid generation. The generation module allows the generation of surface and volume grids based on NURBS generation algorithms. The NURBS database adopted in CAGI allows the geometries / grids to be output to the IGES format with entities 126 (NURBS curve) and 128 (NURBS surface). Thus, the grid output is compatible with the needs of the CAD/CAM system. The intuitive graphical user interface provides a user friendly environment for easy and efficient manipulation and generation of the desired sculptured geometry. In summary, the algorithms developed and incorporated using CAGD techniques in CAGI have demonstrated a significant time saving involved in surface preparations associated with grid generation. Also, CAGI now provides a two–way linkage between the CAD and grid systems.

The CAGD techniques have been utilized in CAD/CAM systems for decades, many of them are well developed and documented in the literature.

Some of these algorithms are adopted in this study. These algorithms include the (NURBS) knot insertion, degree elevation, splitting / joining algorithms [Ref 42~47,76,77] and the evaluation of the normalized BSpline basis functions [Ref 18]. These algorithms have been realized and implemented in this research. Also, the data reduction programs [Ref 38] and the *FORMS* Library [Ref 39], available in the public domain are implemented. Applications of these existing algorithms make the grid generation process easier. For example, applying the perturbations to the NURBS control net makes the simulation of temporally deforming problem efficient. The modeling of volume grid using NURBS control polygons reduces the memory storage requirement.

The grid generation process generally has a more restrictive requirement for modeling the geometry than that of CAD/CAM system. For example, the surface of revolution option for grid generation is not always for a full revolution (360^0 revolution). Also the generation of the conic arc usually requires different rotation angle and the knowledge of the semi–major and semi–minor axis information. The generation of a circular arc may require the different staring and ending angles. Hence, the transformation algorithms for circular arc, conic arc, surface of revolution to NURBS (described in Chapter Two) are all enhanced and generalized for the grid generation procedures. Also, the grid adaptation algorithm [Ref 83~85] is enhanced by replacing the interpolation algorithm with the new reparameterization algorithm (described in Chapter Four) for the precise grid distribution control. When the NASA IGES committee defined the NASA IGES specification, the committee accepted various suggestions from this study (such as the exclusion of the IGES plane entity, entity number 108, from the specification).

The development of many new algorithms related to NURBS is the key contribution of this study. These new algorithms include the transformation of superelliptic arc to a NURBS curve, the transformation of cubic curve (surface) to a BSpline curve (surface) and the transformation of a composite curve to one NURBS curve (described in Chapter Two). Another contribution is related to the generation algorithms. For example, the generation of a cascading surface by NURBS and various NURBS volume generating functions (described in Chapter Three) have not been found in related literature. The new NURBS evaluation algorithm presented in Chapter Four provides a competitive alternative to the de Boor algorithm (in terms of memory and computational time). The new reparameterization algorithm described in Chapter Four already received positive feedback after it was published in the Journal of *Computer Aided Design* [Ref 94].

Considerable advantages have been realized in the development of these algorithms as demonstrated in this study. However, various research issues remain to be addressed. They are described as follows.

Even though the re–parameterization allows the user to control the grid points (lines) distributed along the NURBS entity in the physical space, in many CFS simulations, especially for the viscous and turbulent problems, this distribution function may not be adequate. The orthogonal grid near the boundaries needs to be considered. Since the NURBS is an algebraic method, the orthogonality is an important issue for further study.

The unstructured grid and the hybrid grid techniques have become popular and important in today's CFS simulation. These two techniques provide the flexibility to construct a complex configuration easily. As the examples shown in chapter three, one can generate the unstructured / hybrid grids in

the 2D parametric domain and then utilize NURBS parametric property to evaluate the final 3D unstructured / hybrid grid in the physical space. However, the parametrization problem occurs. Even though the behaviors (aspect ratio, skewness ...) of the triangles or hybrid grids are satisfactory in parametric space, this does not necessarily result in the same grid quality in the physical space. The re–parameterization algorithm for structured grids described in chapter four will not be applicable to the unstructured (triangulated) grid. Hence, the NURBS re–parameterization algorithm for unstructured and hybrid grid generation is an important issue for further research.

Another open question is related to the NURBS deforming geometry. As is described in chapters two and five, NURBS is composed of the control points, *orders*, knot vectors and *weights*. Changing any of this information will lead to a new NURBS entity. However, the difficulty lies in the fact that for certain geometries, there is not sufficient information to model them by NURBS exactly. For example, exact modeling of the sine wave or a bell–shaped curve with concise NURBS control polygons still has not been formulated. For the deforming geometry problems, if the deforming shapes are specified within different time steps, for example, a semi–circular change to a sine wave within time interval 0 ~ 0.5 sec, and from the sine wave to a bell–shaped curve within the time interval 0.5 ~ 1.0, the NURBS modeling technique may fail and a new combination change of the NURBS information must be found.

A final important issue relates to the automatic structured grid generation for the trimmed surfaces defined in the IGES file. The strategies for handling the structured trimmed surface presented in this study allows the engineers to interactively generate the desired structured grid around the

trimmed region. However, in some of the industrial applications, an IGES file may contain thousands of trimmed surfaces. It is necessary to enhance the procedures so that the overall structured grid generation on the trimmed surface can be reduced.

Although there is still room for further work, as presented, CAGI has already successfully bridged the gap between grid systems and CAD/CAM systems, and it has been effectively used in industrial applications. The NURBS volume generation and manipulation algorithms will be extended to enhance the CAGI to be a solid modeling package in the future work.

REFERENCES

[1] Anderson D. A., Tannehill J.C. and Pletcher R. H. "Computational Fluid Mechanics and Heat Transfer", Hemisphere Publishing Corporation, 1984.

[2] Arabshashi, A., "A Dynamic Multiblock Approach to Solving the Unsteady Euler Equations about Complex Conffigurations.", Ph.D Dissertation, Mississippi State University, May 1989.

[3] Barnhill R.E, "Geometry Processing for Desing and Manufacturing", SIAM, 1992.

[4] Barnhill R.E and Riesenfeld R.F., "Computer Aided Geometric Design", Academic Press, Inc., 1974.

[5] Barry J., Wang W. and Cheng F., "Reduced–knot NURBS representations of rational G^1 composite Bezier curves", *Computer–Aided Design*, Volume 26 Number 5, May 1994.

[6] Blake M.W.,Kerr P.A., Thorp S.A., and Chou J.J., "NASA Geometry Data Exchange Specification for Computational Fluid Dynamics — (NASA IGES)". NASA Reference Publication 1338., April, 1994.

[7] Boehm W. and Prautzsch H., "Geometric Concepts for Geometric Design", A K Peters, 1994.

[8] Boehm, W. – "Inserting New Knots into B–Spline Curves", *Computer Aided Design*, Vol. 12, No 4, pp 199~201, July 1980.

[9] Boehm W., Farin G. and Kahmann J.– "A survey of curve and surface methods in CAGD", Computer Aided Geometric Design, pp 1~60, 1984.

[10] Chou J. J. and Piegl L., "Data Reduction Using Cubic Rational B-Splines.", IEEE Computer Graphics & Applications., pp 60~68.

[11] Craft P. E., "A CFD Analysis of a Full Configuration Fighter Aircraft", Master Thesis, Mississippi State University. 1994.

[12] Chan, W.M. and Steger, J.L., "A Generalized Scheme for Three–Dimensional Hyperbolic Grid Generation", AIAA 91–1588, AIAA 10th CFD Conference, Honolulu, 1991.

[13] Boyalakuntla K. "Computational Field Simulation of Temporally Deforming Geometries," Master's Thesis, Mississippi State University, December 1995.

[14] Cohen, E., Lyche. T., and Schumaker, L.L. – "Algorithms for Degree-Raising of Splines", *ACM Transactions on Graphics*, Vol. 4, No 3, pp 171–181, July 1985.

[15] Coons, S.A., "Surfaces for Computer–Aided Design of Space Forms", Tech. Report MAC–TR–41, MIT, Cambridge, Mass, 1967.

[16] Cooper, G.R. and Sirbaugh, JR., "The PARC Distinction: A practical Flow Simulator.", *AIAA/ASME/ SAE/ ASEE* 26th Joint Propulsion Conference (Orlando, FL.) July, 1990. AIAA–90–2002.

[17] Cordova, J.Q., "Advances in Hyperbolic Grid Generation", 4th International Symposium on Computational Fluid Dynamics, September 1991, Davis, CA.

[18] de Boor, C., – *A Practical Guide to Splines*, Applied Mathematical Sciences, Vol 27, Springer–Verlag,1972.

[19] Dennis J.E., and Schnabel R.B., "Numerical Methods for Unconstrained Optimization and Nonlinear Equations", Prentice–Hall, Inc,. 1983.

[20] Dorrell, E.W., and McClure, M.D., "3D INGRID: Interactive Three–Dimensional Grid Generation", AEDC–TR–87–40.

[21] "DT_NURBS Workshop", February 1993, Seattle, Washington.

[22] Fisher S., Heinrich M. and Maitz K., "Graphics Library Programming Guide", Silicon Graphics, Inc., 1989.

[23] Farin, G.E. *Curves and Surfaces for Computer Aided Geometric Design: A Pratical Guide*, Third Edition, Academic Press, 1993.

[24] Farin, G.E. *Geometric Modeling: Algorithms and New Trends.*, SIAM publication, 1987.

[25] Farin, G.E. *NURBS for Curve and Surface Design*, SIAM Publication, 1991.

[26] Forrest, R.A., "Curves and Surfaces for Computer–Aided Design", Ph.D. Dissertation, Cambridge University, Cambridge, UK, 1968.

[27] Foley J.D., Dam A.V., Feiner S.K. and Hughes J.F., "Computer Graphics: Principles and Practice", second edition in C., Addison–Wesley Publishing Company, 1995.

[28] Jones, G.A., "*Surface Grid Generation for Composite Block Grids*", PhD Dissertation, Mississippi State University, May 1988.

[29] Gaither A., Gaither K, Jean B., Remotigue M., Whitmire J., Soni B., Thompson J., Dannenhoffer J and Weatherill N., "The National Grid Project: A System Overview", Proceeding of "Surface Modeling, Grid Generation, and Related Issues in Computational Fluid Dynamic (CFD) Solutions. pp 423~446, May 1995.

[30] Goldman R.N. and Lyche Tom, "Knot Insertion and Deletion Algorithms for B–Spline Curves and Surfaces", SIAM, 1993.

[31] Hagen H., "Topics in Surface Modeling", SIAM Publication, 1992.

[32] Hagen H., "Curve and Surface Design", SIAM Publication, 1992.

[33] Hamman, B, "NURBS in Grid Generation", Note for grid generation short course (unpublished), Mississippi State University, 1992.

[34] Hoschek J. and Lasser D., "Fundamentals of Computer Aided Geometric Design", A K Peters, 1989.

[35] Initial Graphics Exchange Specification (IGES) Version 5.1 , distributed by the National Computer Graphics Association (NCGA), Technical Services and Standards, IPO Administrator, 2722 Merrilee Drive, Suite 200, Fairfax, VA 22031.

[36] Ives, D., Miller R., Siddons W., and Dyke V. K., "Grid Generation – A View from the Trenches.", Proceedings of NASA Workshop on Surface modeling, Grid Generation, and Related Issues in CFD solutions, pp 45~55. Cleveland, Ohio. May 9~11, 1995.

[37] Lin F. and Hewitt W.T., "Expressing Coons – Gordon surfaces as NURBS.", Journal of Computer Aided Design, Volume 26., Number 2, pp. 145~155. February 1994.

[38] Lyche T. and Morken K., "A Data–Reduction Strategy for Splines with Applications to the Approximation of Functions and Data.", IMA Journal of Numerical Analysis (1988), volume 8., pp 185~208.

[39] Overmars, M.H., "FORMS: A Graphical User Interface Toolkit for Silicon Graphics Workstations", Department of Computer Science, Utrecht University, Utrecht, Netherlands, 1991.

[40] OpenGL Architecture Review Board, "OpenGL Reference Manual", "OpenGL Programming Manual", Addison–Wesley Publishing Company., 1992.

[41] Pao S. P., Carlson J. R. and Abdol–Hamid K. S., "Computational Investigation of Circular–to–Rectangular Transition Ducts.", Journal of Propulsion and Power., Vol 10., No 1., pp95~100. Jan–Feb 1994.

[42] Piegl, L. and Tiller, W.– "Curve and surface constructions using rational B–splines", Computer Aided Design, Vol 19, NO 9, 1987, pp 485~498.

[43] Piegl, L. - "Rational B-Spline Curves and Surfaces for CAD and Graphics", *State of the Art in Computer Graphics, Visualization and Modeling,* Eds, David F. Rogers, Rae A. Earnshaw, Springer-Verlag, pp 225-269, 1991.

[44] Piegl, L., - "On NURBS: A survey", *IEEE Computer Graphics & Applications,* Vol 11, No 1, pp 57 - 71, January, 1991.

[45] Piegl, L. and Tiller, W.- "A Menagerie of Rational B-Splines", IEEE Computer Graphics & Applications, Vol 9, NO 5, Sep 1989, pp 48-56.

[46] Piegl, L. - "Infinite Control Points-A Method for Representing Surfaces of Revolution Using Boundary Data", *IEEE Computer Graphics & Applications,* Vol. 7, No. 3,pp 45-55, March 1987.

[47] Piegl, L. - "Interactive Data Interpolation by Rational Bezier Curves", *IEEE Computer Graphics & Applications,* Vol 7 No 4, pp 45-58, April 1987.

[48] Reichert B. A., Hingst W. R. and Okiishi T. H., "Circular-to-Rectangular Transition Duct Flow Without and with Inlet Swirl.", Journal of Propulsion and Power., Vol 10., NO. 1., pp 88-100. Jan-Feb. 1994.

[49] Riesenfeld, R.F., "Applications of B-Spline Approximation to Geometric Problems of Computer-Aided Design", Ph.D. Dissertation, Syracuse University, Syracuse, N.Y., 1973.

[50] Rogers D.F. and Adams J.A., "Mathematical Elements for Computer Graphics", Second edition, McGraw-Hill Publishing Company, 1990.

[51] Shih, M.H., Yu, T.Z. and Soni, B., "Interactive Grid Generation and NURBS Applications," *Journal of Applied Mathematics and Computations,* Volume 65, Numbers 1-3, September 1994.

[52] Shih M. H, Soni B. K., Yu T.Y , and Shaunak S.,"TIGER: A Turbomachinery Grid generation/Simulation System", AIAA - 94 - 0207, the 32nd Aerospace Sciences Meeting & Exhibit, January 10-13, 1994, Reno, Nevada.

[53] Shih, M.H., 'Towards a Comprehensive Computational Simulation System for Turbomachinery,' Ph.D. Dissertation, Mississippi State University, May 1994.

[54] Shih, M.H. and Soni, B.K.,"Grid Generation for 3D Turbomachinery Configurations", AIAA-92-3671, AIAA/SAE/ASME/ASEE 28th Joint Propulsion Conference, Nashville, TN, July 1992.

[55] Shih, M.H. and Soni, B.K., "Geometry Modeling and Multi-Block Grid Generation For Turbomachinery Configurations", Proceedings of the NASA Workshop on Software Systems for Surface Modeling and Grid Generation, Hampton, VA, April 1992.

[56] Shih M. H., Soni, B.K., Yu, T.Y., and Shaunak S. "TIGER: A Turboma-chinery Grid Generation/Simulation System," AIAA – 94 – 0207, 32nd Aerospace Sciences Meeting & Exhibit., Reno, Nevada January 10–13, 1994.

[57] Soni, B. K., Weatherill, N.P., and Thompson, J.F., "Grid Adaptive Strate-gies in CFD," *Advances in Hydro-Science & Engineering*, University of Mississippi Press, Wang, S. S. Y. (ed). 1.A: pp 201~208, 1993.

[58] Soni, B., Yu, T.Z. and Vaughn, D., "CAGI: Computer Aided Grid Inter-face– A Work in Progress," Proceedings of the Tenth Workshop for Com-putational Fluid Dynamics Applications in Rocket Propulsion, pp 577 ~ 614, NASA George C. Marshall Space Flight Center, Huntsville, AL, April 1992.

[59] Soni, B.K., Yang S., "NURBS Based Surface Grid Redistribution and Re-mapping Algorithms," Computer Aided Geometric Design 12, pp 675~692. Nov 1995.

[60] Soni B. K., "Grid Generation for Internal Flow Configurations," *Journal of Computers & Mathematics with Applications*, Vol. 24, No. 5/6, pp. 191 ~ 201, September 1992.

[61] Soni, B.K., "GENIE: GENeration of Computational Geometry–Grids for Internal–External Flow Configurations", Proceedings of Numerical Grid Generation in Computational Fluid Mechanics (88), Pineridge Press, pp. 915–924, 1988.

[62] Soni, B.K., and Dorrell, E.W., "INGRID Interactive Geometry–Grid Gen-eration for Two Dimensional Applications", AEDC–TR–86–49.

[63] Soni, B.K., McClure, M.D., and Mastin, C.W., "Geometry and Generation in N+1 Easy Steps", The First International Conference on Numerical Grid Generation in Computational Fluid Dynamics, Landshut, W. Ger-many, July 1986.

[64] Soni, B.K., Thompson, J.F., Stokes, M., and Shih, M.H., "GENIE++, EA-GLEView and TIGER: General and Special Purpose Graphically Interac-tive Grid Systems", 30th Aerospace Sciences Meeting & Exhibit, AIAA – 92–0071, Reno, Nevada, January 1992.

[65] Soni, B.K. and Shih, M.H., "TIGER: Turbomachinery Interactive Grid GenERation", Proceedings of the Third International Conference of Nu-merical Grid Generation in CFD, Barcelona, Spain, June 1991.

[66] Soni, B.K. and Shih, M.H., "TURBOGRID: Turbomachinery Applications of Grid Generation", 26th AIAA/SAE/ASME Joint Propulsion Confer-ence, AIAA–90–2242, Orlando, Florida, July 1990.

[67] Soni B.K., Thornburg H.J., Shih M. H., Yu T.Y, Craft P.E. and Shaunak S., "Prediction of Three–Dimensional Flow About Complex Geometries" Proceeding of High Performance Computing, pp 126~131, April 1994.

[68] Steger, J.L. and Chaussee, D.S., "Generation of Body–Fitted Coordinates Using Hyperbolic Partial Differential Equations", SIAM J. Sci. Stat. Computation, Vol. 1, No. 4, Dec. 1980.

[69] Strikwerda J. C., "Finite Difference Schemes and Partial Differential Equations", Wadsworth & Brooks/Cole Advanced Books & Software, 1989.

[70] Straber W. and Seidel H.P., "Theory and Practice of Geometric Modeling", Springer–Verlag, 1989.

[71] Thompson, J.F., "National Grid Project," *Computing Systems in Engineering*, Vol. 3, Nos. 1~4, pp. 393~399, 1992.

[72] Thompson, J. F., "A survey of grid generation techniques in computational fluid dynamics", AIAA–83–0447, AIAA 21st Aerospace Sciences Meeting.

[73] Thompson, J F, Warsi , Z.U.A., Mastin C.W., *Numerical Grid Generation Foundations and Applications*, North–Holland, 1985.

[74] Thompson, J.F., "Program EAGLE–Numerical Grid Generation System User's Manual", Vols. II, III, AFATL–TR–87–15, March 1987.

[75] Thompson, J.F., "A General Three–Dimensional Elliptic Grid Generation System On A Composite Block Structure", Computer Methods in Applied Mechanics and Engineering, 1987.

[76] Tiller, W.– "Knot Removal Algorithms for NURBS Curves and Surface," *Computer Aided Design*, Vol 24, NO 8, pp 445~453. August 1992.

[77] Tiller, W. – "Rational B–Splines for Curve and Surface Representation", *IEEE Computer Graphics & Applications*, Vol. 3, No 10, pp 61~69, Sep 1983.

[78] Versprille, K.J., "Computer–Aided Design Applications of the Rational B–Spline Approximation Form", Ph.D. Dissertation, Syracuse University, Syracuse, N.Y., 1975.

[79] Watkins D.S., "Fundamentals of Matrix Computations", John Wiley & Sons., 1991.

[80] Watt A. and Watt M., "Advanced Animation and Rendering Techniques: Theory and Practice", Addison–Wesley Publishing Company, 1992.

[81] Yoon Y. H.,"Enhancements and Extensions of Eagle Grid Generation System", Ph. D. dissertation, Mississippi State University, 1991.

[82] Yoon Y. H.,"Optimal Domain Decomposition Strategies", Proceedings of NASA Workshop on Surface modeling, Grid Generation, and Related Issues in CFD solutions, pp 597~613. Cleveland, Ohio. May 9~11, 1995.

[83] Yang, J. C.,"General Purpose Adaptive Grid Generation System", Ph. D. dissertation, Mississippi State University, 1993.

[84] Yang, J. C., and Yu T. Y., "The Application of NURBS in Adaptive Grid Generation", Proceeding of International Mathematics Conference '94, National Sun Yat–Sen University, Kaohsiung, Taiwan, Republic of China, Dec. 2~5, 1994.

[85] Yang J. C. and Yu, T. Y., "General Purpose Adaptive Grid Generation Using Redistribution Scheme for Steady and Unsteady Simulated Flows", proceeding of the 3rd Computational Fluid Dynamic Conference, pp 489~496. Taiwan, R.O.C., Aug 19~20, 1995.

[86] Yu T.Z. "IGES Transformer and NURBS in Grid Generation," Master's Thesis, Mississippi State University, August 1992.

[87] Yu, T.Z. and Soni, B.K., "IGES Transformer and NURBS in Grid Generation," Proceedings of the Eleventh Workshop for Computational Fluid Dynamics Applications in Rocket Propulsion, pp 1021~1053, NASA George C. Marshall Space Flight Center, Huntsville, AL, April 1993.

[88] Yu T. Y. and Soni B. K., "Geometry Transformer and NURBS in Grid Generation", Proceedings of the 4th International Conference on Numerical Grid generation in Computational Fluid Dynamics and Related Fields. (Eds) N.P. Weatherill, P.R. Eiseman, J. Hauser, and J. F. Thompson, P 353~364. Conference held at Swansea, Wales (UK), April 6~8. 1994.

[89] Yu, T.Z. and Soni, B.K, "CAGI: Computer–Aided Grid Interface (CAGI Version 1.0)" Proceedings of the 12 Workshop for Computational Fluid Dynamics Applications in Rocket Propulsion, NASA George C. Marshall Space Flight Center, Huntsville, AL, April 1994.

[90] Yu T. Y. and Soni B. K., "CAGI: Computer Aided Grid Interface", AIAA – 95–0243, the 33rd Aerospace Sciences Meeting and Exhibit, Reno, NV, January 9~12 1995.

[91] Yu T. Y. and Soni B. K. ,"Geometry Modeling and Grid Generation Using 3D NURBS Control Volume", Proceedings of NASA Workshop on Surface modeling, Grid Generation, and Related Issues in CFD solutions, pp 491~503. Cleveland, Ohio. May 9~11, 1995.

[92] Yu Tzu–Yi, Soni B.K. and Yang J. C., "A Memory–Saving Strategy in grid Generation for the CFD Applications", proceeding of the 3rd Computational Fluid Dynamic Conference, pp 489~496. Taiwan, R.O.C., Aug 19~20, 1995.

[93] Yu T. Y. and Soni B. K. , "Surface and Volume grid generation in parametric form", Proceedings of NASA Workshop for Computational Fluid Dynamic (CFD) Applications in Rocket Propulsion and Launch Vehicle Technology, April 25~27, 1995.

[94] Yu T. Y. and Soni B. K., "The Application of NURBS in numerical grid generation", *Journal of Computer–Aided Design*, Volume 27., Number 2, pp. 147~157. February 1995.

APPENDIX A

AN IGES FILE FORMAT

```
This file was produced by CAGI (Computer Aided Grid Interface) beta1.0        S    1
„19HERC/NSF Miss. State,10Hsphere.igs,                                         G    1
17HMSU/ERC CAGI V1.0,19HCAGI V1.0, 10/10/94,32,38,6,                           G    2
308,15,11HUnspecified,1,1,4HINCH,32,32,13H951109.173219,0.000001,             G    3
2.00000000000000,8HRobertYu,11HNSF/ERC/MSU,9,0,13H951109.173219;              G    4
    . 128    1    0    0    0    0    0                    0        00000000D   1
      128    0    0   52    0    0    0            B_SPL_SF           1D    2
128, 4, 8, 2, 2, 0, 0, 0, 0, 0,                                        1P    1
0.000000E+00, 0.000000E+00, 0.000000E+00, 5.000000E-01,               1P    2
5.000000E-01, 1.000000E+00, 1.000000E+00, 1.000000E+00,               1P    3
0.000000E+00, 0.000000E+00, 0.000000E+00, 2.500000E-01,               1P    4
2.500000E-01, 5.000000E-01, 5.000000E-01, 7.500000E-01,               1P    5
7.500000E-01, 1.000000E+00, 1.000000E+00, 1.000000E+00,               1P    6
1.000000E+00, 7.071068E-01, 1.000000E+00, 7.071068E-01,               1P    7
1.000000E+00, 7.071068E-01, 5.000000E-01, 7.071068E-01,               1P    8
5.000000E-01, 7.071068E-01, 1.000000E+00, 7.071068E-01,               1P    9
1.000000E+00, 7.071068E-01, 1.000000E+00, 7.071068E-01,               1P   10
5.000000E-01, 7.071068E-01, 5.000000E-01, 7.071068E-01,               1P   11
1.000000E+00, 7.071068E-01, 1.000000E+00, 7.071068E-01,               1P   12
1.000000E+00, 7.071068E-01, 5.000000E-01, 7.071068E-01,               1P   13
5.000000E-01, 7.071068E-01, 1.000000E+00, 7.071068E-01,               1P   14
1.000000E+00, 7.071068E-01, 1.000000E+00, 7.071068E-01,               1P   15
5.000000E-01, 7.071068E-01, 5.000000E-01, 7.071068E-01,               1P   16
1.000000E+00, 7.071068E-01, 1.000000E+00, 7.071068E-01,               1P   17
1.000000E+00, 1.000000E+00, 0.000000E+00, 0.000000E+00,               1P   18
1.000000E+00, 1.000000E+00, 0.000000E+00,-4.371139E-08,               1P   19
1.000000E+00, 0.000000E+00,-1.000000E+00, 1.000000E+00,               1P   20
0.000000E+00,-1.000000E+00,-8.742278E-08, 0.000000E+00,               1P   21
1.000000E+00, 0.000000E+00, 0.000000E+00, 1.000000E+00,               1P   22
1.000000E+00, 1.000000E+00,-4.371139E-08, 1.000000E+00,               1P   23
1.000000E+00,-1.000000E+00, 1.000000E+00, 1.000000E+00,               1P   24
-1.000000E+00,-8.742278E-08, 0.000000E+00, 1.000000E+00,               1P   25
0.000000E+00, 0.000000E+00, 1.000000E+00,-4.371139E-08,               1P   26
1.000000E+00,-4.371139E-08,-4.371139E-08, 1.000000E+00,               1P   27
-1.000000E+00,-4.371139E-08, 1.000000E+00,-1.000000E+00,               1P   28
-8.742278E-08, 0.000000E+00, 1.000000E+00, 0.000000E+00,               1P   29
0.000000E+00, 1.000000E+00,-1.000000E+00, 1.000000E+00,               1P   30
-4.371139E-08,-1.000000E+00, 1.000000E+00,-1.000000E+00,               1P   31
-1.000000E+00, 9.999999E-01,-1.000000E+00,-8.742278E-08,               1P   32
0.000000E+00, 1.000000E+00, 0.000000E+00, 0.000000E+00,               1P   33
1.000000E+00,-1.000000E+00,-8.742278E-08,-4.371139E-08,               1P   34
-1.000000E+00,-8.742278E-08,-1.000000E+00,-1.000000E+00,               1P   35
-8.742278E-08,-1.000000E+00,-8.742278E-08, 0.000000E+00,               1P   36
1.000000E+00, 0.000000E+00, 0.000000E+00, 1.000000E+00,               1P   37
-1.000000E+00,-1.000000E+00,-4.371139E-08,-9.999999E-01,               1P   38
-1.000000E+00,-1.000000E+00,-9.999999E-01,-1.000000E+00,               1P   39
-1.000000E+00,-8.742278E-08, 0.000000E+00, 1.000000E+00,               1P   40
0.000000E+00, 0.000000E+00, 1.000000E+00, 1.311342E-07,               1P   41
```

-1.000000E+00,-4.371139E-08, 1.311342E-07,-1.000000E+00,   1P 42
-1.000000E+00, 1.311342E-07,-1.000000E+00,-1.000000E+00,   1P 43
-8.742278E-08, 0.000000E+00, 1.000000E+00, 0.000000E+00,   1P 44
0.000000E+00, 1.000000E+00, 1.000000E+00,-9.999999E-01,   1P 45
-4.371139E-08, 1.000000E+00,-9.999999E-01,-1.000000E+00,   1P 46
1.000000E+00,-9.999998E-01,-1.000000E+00,-8.742278E-08,   1P 47
0.000000E+00, 1.000000E+00, 0.000000E+00, 0.000000E+00,   1P 48
1.000000E+00, 1.000000E+00, 1.748456E-07,-4.371139E-08,   1P 49
1.000000E+00, 1.748456E-07,-1.000000E+00, 1.000000E+00,   1P 50
1.748456E-07,-1.000000E+00,-8.742278E-08, 0.000000E+00,   1P 51
0.000000E+00, 1.000000E+00, 0.000000E+00, 1.000000E+00;   1P 52
S   1G   4D   2P   52   T   1

APPENDIX B

NURBS SURFACE FORMAT

```
#=================================================
#The number of control points of the surface
5 2
#The two degrees of the surface
2 1
#The two knot vectors
0.000000
0.000000
0.000000
0.500000
0.500000
1.000000
1.000000
1.000000
0.000000
0.000000
1.000000
1.000000
#The coordinates of the control points and weight
-1.950000 0.000000 0.000000 1.000000
-1.950000 1.200000 0.000000 0.707107
0.000000 1.200000 0.000000 1.000000
1.950000 1.200000 0.000000 0.707107
1.950000 0.000000 0.000000 1.000000
-8.000000 -0.000001 0.000000 1.000000
-8.000001 8.000000 0.000000 0.707107
0.000000 8.000000 0.000000 1.000000
8.000000 8.000000 0.000000 0.707107
8.000000 0.000000 0.000000 1.000000
```

Computer-Aided Grid Interface (CAGI)
User's Manual
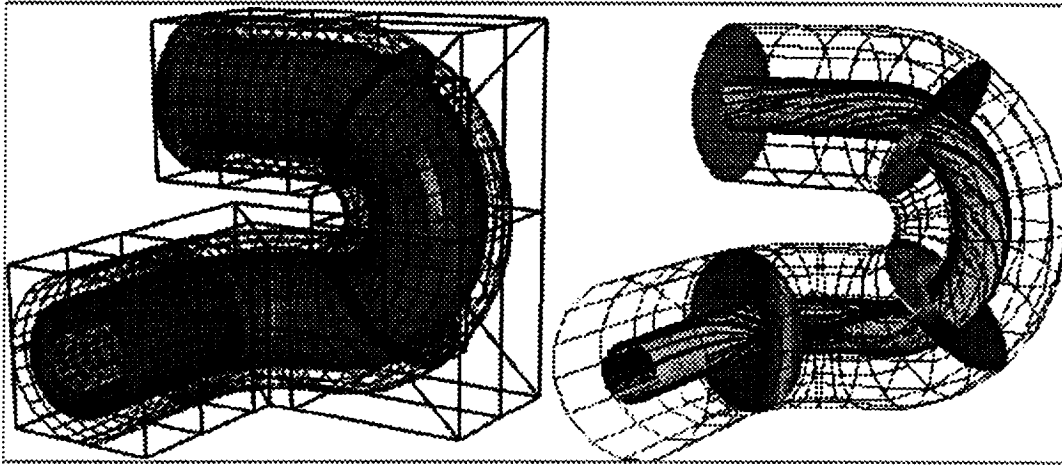
Version 1.0

**Mississippi State University / National Science Foundation
Engineering Research Center for Computational Field Simulation**

Program Developed by Dr. Tzu-Yi (Robert) Yu
Documentation by Dr. Tzu-Yi (Robert) Yu and Crystal Jordan

Table of Contents

Please click on the hypertext to view the referenced documents.

Current Authors:

Crystal Jordan

Dr. Tzu-Yi "Robert" Yu

Introduction

Why is CAGI needed?

The computational Field Simulation (CFS) for physical problems has become more practical due to the progress made in computer memory availability and the computer speed. This CFS process generally involves the steps:

1. Pre-processing: includes numerical grid generation, boundary condition set up and definition involving sculptured geometry description of all solid components and field boundaries / surface.
2. Processing: Numerical solving a set of non-linear partial differential equations at the grid point developed by a pre-processor.
3. Post-processing--Numerical visualization of flow field properties simulated by the solution processor.

In this process, the sculptured modeling and the grid generation are the most time consuming and laborious work. It is necessary to develop a system which can reduce the geometry contruction and grid generation time for the CFS process.

For a routine application of CFS, in an industrial environment, the overall response time for CFS must be reduce considerably. As noted by several scientists:

" ...The industrial requirement is for reliable one hour grid generation turnaround for one-time geometries when run by designers. The system must include CAD-to-Grid links which resolve tolerance issues and produce grids with a quality good enough for the flow solver. The designer has to feel that the grid generation processes is under control and is predictable."

--- David Ives, Robert Miller, William Siddons and Kevin VanDyke. May 1995

In many of today's industrial applications, most of the geometrical configurations of interest to practical problems are designed using a CAD/CAM system. Setting up the communication between CAD/CAM design tools or other grid generation system would be the ideal situation. And this idea was pointed out by the NASA Steering Committee on Surface Modeling and Grid Generation on 1992. Unfortunately, the CAD/CAM systems have many different geometry output formats which force the designer to spend a gread deal of time manipulating geometrical entitites in order to achieve a useful sculptured geometrical description for grid generation. In addition, there is a danger of losing the fidelity of the geometry in this process of data transfer between different Input/Output (I/O) formats. The other issure related field simulation is the grid quality. The quality of the grid affects the accuracy of the solution and the computation time. It may be necessary to reconstruct the grids for a more satisfactory result after obtaining the first solution. This reconstruction procedure involves a change of either resolution (the size of the grids) or the spacing (the distance between grid points) functins. However, this proess is tedious and very time consuming; this is especially true for a complex geometry.

To bridge the gap between the CAD/CAM systems and the grid generation systems, it is necessary to establish communicatin paths so that the geometries and grids defined within these two systems can be linked with each other. For most of the CAD/CAM systems, the Initial Graphics Exchange Specificatin (IGES) is a widely accepted standard for geometry exchange. Most CAD/CAM systems support the IGES format as an Input/Output of resulting geometries. And for the grid generation part, there are many approaches for representing sculptured geometry/grid, such as the Bezier curve/surface, parametric cubic curve/surface, Hermit representation, Transfinite Interpolation (also referred as TFI), ... and so on. Among these representations, the Non Uniform Rational BSpline (NURBS) has been widely utilized in many applications. NURBS is getting popular because it has a powerful features, such as the local control property, variation diminishing and convex hull, .. etc. Also, the geometry tool kits, like the curve/surface interpolation, data reduction, degree elevation, knot insertion and splitting are all well-developed. These features make the **NURBS** very useful not only in the CAD/CAM packages but also in Grid Generation systems. And what important is -- the **IGES** file already included the **NURBS** curve (entity 126) and surface (entity 128). Therefore, if a software package can read in an **IGES** file, and convert any Non **NURBS** entities to **NURBS** curve/surface, then the geometry defined in **IGES** can be utilized by grid system. Or in the other way, if the geometry/grid defined in grid system is represented as **NURBS**, then it is possible to output the grids to an **IGES** format with entity 126 and 128. If this can be done, then the geometry and grid are communitable, and the construction time for the pre-posser can be reduced. CAGI is aimed at this goal. It is a package with NURBS database. It can read in the **IGES** file and transforms the geometrical definitions to **NURBS**, or define the grids with **NURBS** representation and output the grid with **IGES** data.

What is CAGI?

CAGI stand for Computer-Aided Grid Interface. It is a grid genration package with **NURBS** database. The Graphic User Interface (GUI) is made by utilizing the *FORM Library*, and the *SGI Graphics Library* is utilized for the graphic display. **CAGI** can either read the standard **IGES** format or generate grids form NURBS definition. The representation of **NURBS** has been extended from curve (1D), surface (2D) to volume (3D) definition. This project is sponsored by the NASA Marshall space flight center. Any suggestion or request can be forwarded to the technical monitors, Mr. Ted Benjamin and Robert Williams.

Currently,**CAGI** contains several modules:

     O **Geometry Generation Module**
        This module allows the user to generate the grids from NURBS definition. The user can create points, curves, surfaces or volume by the different NURBS options.

     O **Geometry Manipulation Module**
        This Module allows the user to manipulate the selected geometric entities (either curves, surfaces or volume). The user is allowed to change the orentation, the resolutions, or even the distribution of the selected entity easily by clicking the proper button or slider bars. Since **CAGI** transforms the geometry to the NURBS definition, this module takes advantage of NURBS properties/tools such as knot insertion, degree elevation, data reduction and even the alternation of the location of control polygon and weight to change the shape of the geometry.

o Volume Grid Module & Geometry Viewing
These two modules are temperatly not fully function. They are under construction.

Contact Information

General Contact:

Dr. Bharat K. Soni
P. O. Box9627
Mississippi State, MS 39762-9627
(601)325-2647 or (601)325-8278

bsoni@erc.msstate.edu

Technical Problem or CAGI Question:

Dr. Tzu-Yi (Robert) Yu
P. O. Box 9627
Mississippi State, MS 39762-9627
Tel:(601)325-2467, Fax:(601)325-7692

yu@erc.msstate.edu

CAGI request:

Mr. Ted Benjamin
ED32, NASA Marshall Space Flight Center
Huntsville, AL 35812
Tel:(205)544-9402, Fax:(205)544-1215

tedb@tyrell.msfc.nasa.gov

Questions or comments about this document:

crystal@erc.msstate.edu

Organization of the CAGI Manual

In this manual, you will find descriptions of the commands in CAGI and instructions on how to use them. You will also find tutorials that give you step-by-step instructions on some examples.

The manual is arranged in the following manner:
- O *CAGI Basics* introduces the **CAGI** Interface.
- O *CAGI Men Buttons* describes the **CAGI** options and commands.
- O *Tutorials/Demos* help you get started learning the functions of CAGI.
- O *Reference* allows you to find the information you need more easily.

Navigating the CAGI Manual

In addition to hyperlinked texts, there are navigation buttons at the end of each document. The navigation buttons are described as follows:

takes you to the Table of Contents

takes you to Index

takes you to the previous file in linear sections of the document; takes you to the parent file in non-linear sections

takes you to the beginning of the Manual

takes you to the next file in in linear sections of the document; takes you to the first child in non-linear sections

CAGI Basics

● Starting CAGI

   O **Limition:** **CAGI** is implemented by C language, the memory allocation function is used. Unless the user runs a very big **IGES** file or create several huge grids to consume the memory, otherwise, otherwise, it can be executed in a small box like *Personal IRIS*. However, since the Graphics functions and the *GUI* are implemented by utilizing the *SGI* Graphics library, this program can only be executed on *SGI* machine which supports the *GL* library.

   O **Execution:** To run **CAGI**, simply type "CAGI &".

● CAGI Main Window

This image shows an orientation of CAGI Screen Layout. It contains the geometry displayed in the graphics window and the entire GUI layout with several main modules. Click here for a large image. For each general section of the CAGI is discussed in detail as follows:

- **Entity Name List**
  The **Entity Name List** panel is placed at the left side of CAGI. The panle lists the names of all the entities in the database (either created from an IGES file or generated from the build function). If a geometry (any grid) is created, then it will be displayed in the graphics window with the proper named (with a highlight bar with light blue color) showed in this Entity Name List panel. If an entity name is highlighted with a light blue bar, then that entity is referred as *active* and will be displayed on the graphichs window. If an entity name is not highlighted, the entity is referred as *inactive* and will not be shown in the graphic window. The user can turn on or off entities by using the mouse to highlight or unhighlight an entity's name. For example, the use can put the mouse to any of the entity name listed in this panel, and use the right mouse to click the name to toggle on/off of the entity.

- **Message Window**
  The *Message Window* is placed at the bottom left corner of CAGI. It is designed to show any error, warning or status messages for the user.

- **Transformation Panel**
  The last panel in the main CAGI window is related to the tranformation and graphics window. The *Transformation Panel* provides another options for rotating objects.

O The ▓▓ buttons indicate rotation with respect to the positive axes.

O The ▓▓ buttons indicate rotation with respect to the negative axes.

There are three pairs of rotation buttons available, each pair is coupled with one principal direction (i.e, the X, Y or Z axis). For example, ▓▓ ▓▓ ▓▓ press this rotation button indicate the user like to rotate the active objects with respect to positive or negative X axis. The rotation buttons for the same direction are mutual excluded (either positive or negative direction). The user can select the rotation option for different principal axis at the same time.

O The ▓▓▓▓ button allows the user to Rotate the objects continually once the rotation direction is decided. In other words, after the rotation buttons are selected, press this button will make the geometry objects continually rotated.

The sensitivity of the *Rotation, Translation* and *Scale* may be adjusted by using the buttons found under the Sensitivity portion of the panel:

O The ▓▓ buttons decrease the sensitivity of Translation, Rotation, and Scaling.

O The ▓▓ buttons increase the sensitivity of Translation, Rotation, and Scaling.

The single arrow allows a minor decrease (or increase), while the double arrow button allows a major decrease (or increase).

The Transformation Panel also has functions which allow the user to adjust the center of rotation:

O The ▓▓▓ button causes CAGI to calculate the center of all the active entites and bring the active entities to the center of the graphic window. This button is used when the entities are outside the graphics window after rotation, translation or scaling. CAGI (actually, the GL library) uses the origin by default for transformation. This can be very inconvenientif the entities are far away from the origin. The rotation would be very hard to control if the rotation radius is very huge. The *Reset_V* button counteracts this problem.

O The ▓▓▓ button tells CAGI to use the center of the entites as the rotating center. This provides an each way to view the geometry even when it's far away from the origin. To calculate the current geometric center, press the reset_v button.

O The ▓▓▓ button displaces the xyz-coordinate frame.

This Transformation Panel also allows the user to change the background color of the graphic window.

O The ▓▓▓ button allows the background color of the graphic window to be changed. Once the button is selected, a color panel will appear. The user may change the color by clicking on the desired color. The left and right arrow keys found on either side of the

pop-up screen give more color palettes to from which to choose.

Other functions available in the Transformation Panel include:

o The ▓▓▓▓ button allows the user to take a "snapshot" or picture of the entities displayed in the graphics window. Once the button is pressed, the *snapshot* icon will appear. The user then must:

1. Place the cursor over the *snapshot* icon until the cursor turns into a camera.

   Note: If the snapshot icon appears in an area where it is undesirable, the user can move the icon by placing the cursor over it and selecting the Alt and the F7 keys on the keyboard. The cursor will become crosshairs and the user can move the icon to a spot which is more convenient.

2. Hold down the middle mouse button. This keeps the camera cursor even when the cursor is moved from the *snapshot* icon.

3. Select the right mouse button.

4. Select *Redraw Rubberband* by moving through the menu until it is highlighted and releasing the right mouse button.

5. Select and hold the left mouse button. Move the mouse to draw a rubberband around the entity you wish to take a snapshot of. Once the rubberband is drawn, it may be resized using the horizontal, vertical and corner cursors or moved using the crosshair cursor.

6. Select *New file name* by using the same procedure outlined in step 4. Once the **File:** menu appears, enter the name and save it with either an **.rgb** or **.gif** file extension.

7. Select *Save and Exit* using the same procedure outlined in step 4.
   Note: If the user wishes to take more than one picture, the user can choose **Save as <file>** and then go to steps 4 through 7 for each picture.

   Note: For more information on using SnapShot, please refer to the Man Pages

o The ▓▓▓▓ button makes the graphics window resize to fill the entire CAGI screen. The user may exit this view by pressing the **Esc** key on the keyboard.

o The ▓▓▓▓ button allows the user to delete unwanted entities. Generally, the user can delete the geometric entity by using the button designed in the Geometry Manipulation Panel. Also, the user can turn on/off of an entity by placing the cursor to the Entity Named List Panel and clicking the entity. However, if the user like to delete an object without going to manipulation panel, one can use this button to reach the goal. To utilize this function, press the **DelObj** icon (and the button will keep on pressed). Next pick the entity name from the Entity Name List. To stop deleting objects, click the **DelObj** button again (and the button will be released). After the button is released, the user can use the mouse to click the entities

in the name list panel to turn on/off the objects.

    o The ▨▨▨▨▨ button allows the user to refresh the graphics window.

Note: If users have any suggestions for additional functions, please send e-mail to Dr. Tzu-Yi "Robert" Yu yu@erc.msstate.edu.

● View Manipulation of the Graphic Window

The entities displayed in the graphics window can also be manipulated using the mouse:

o **Translation**
Entities may be translated by holding down the right mouse button and moving the mouse in the desired direction. To move an entity down, hold the right mouse button and move the mouse toward you. To move an entity up, hold down the right mouse button and move the mouse away from you. To move an entity left, hold down the right mouse button and move the mouse to the left. To move an enity to the right, hold down the right mouse button and move the mouse to the right.

o **Rotation**
Entities may be rotated by holding down the left mouse button and moving the mouse in the desired direction.

o **Scaling**
Entities may be scaled by holding down the middle mouse button and moving the mouse in the desired direction. To zoom in on an entity, hold down the middle mouse button and move the mouse toward you. To zoom out, hold down the middle mouse button and move the mouse away from you.

*Note: The sensitivity of these functions is controlled by the **Transformation Panel**. If the user would like to reset the graphics window to the original view, select the **Reset_V** button on the **Transformation Panel**. If the geometric entities are far away from the origin and make a large rotation axis, press the **Reset_v** button first and then press the **Center** button to overcome the problem.*

● CAGI Menu Buttons

CAGI has been designed with six menu buttons which remain on the top screen at all times (unless full screen is chosen). These buttons are File,Set, Graphic,Entity,Help, and Exit buttons.

    o The ▨▨▨▨▨ menu button cause a pop-up menu to appear shown as follows:

The functions of this pop-up menu allow the user to input or output files in various formats. These functions are discussed below:
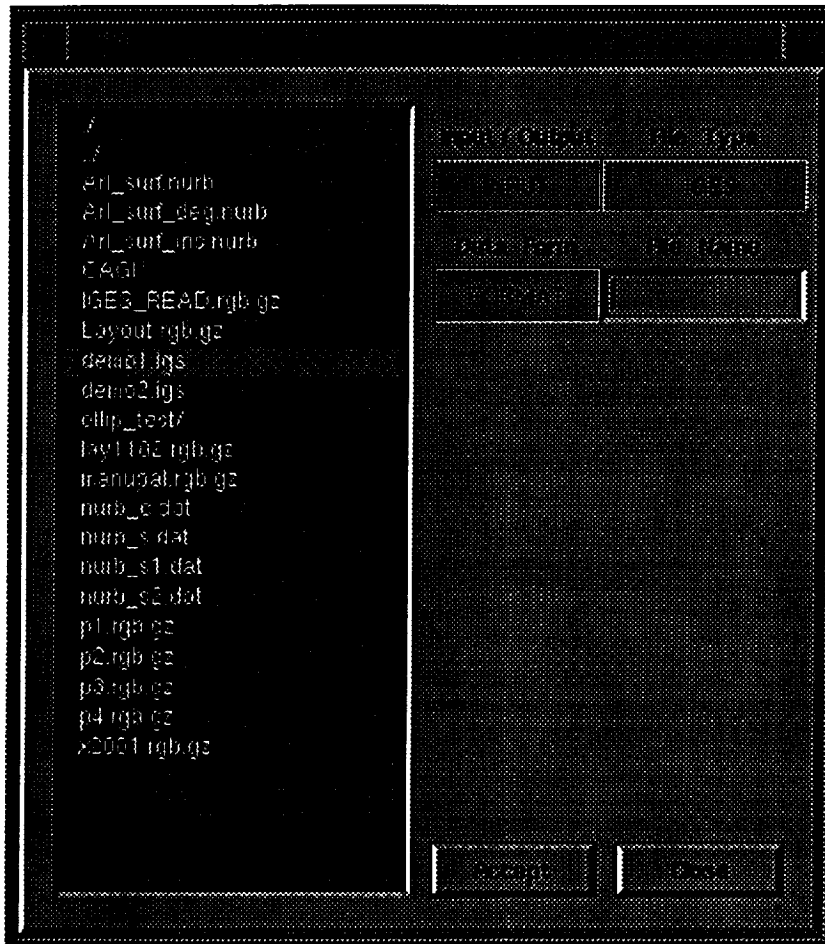
- **Input / Output**

  The *Input / Output* field allows the user to either read in or write out a file. The user can click the button to switch for either Input or Output function.

- **File Type**
  The *File Type* field allows the user to select the file format. The user can select the IGES file format as input or create an output file for all the active entities in either IGES, FAST(Plot3D) or NURBS format.

If a Plot3D file is to be read in, the user should go to the Geometry Generation Module and select either curve or surface for input. For reading the Plot3D file, please refer to Interpolate and Plot3d functions in Geometry Generation Panel.

A Plot3D volume file can only be read after the **Volume Grid Module** has been utilized.

- **Data Form**

  The *Data Form* field allows the user to select either **Formatted** or **Unformatted** data for output.

- **File Name**

  The *File Name* input field allows the user to type in the file name. Or if the user clicks the file name listed in the browser, the selected file name will be displayed in this input field.

  After selecting the proper data formats and keying the file name, the user should press the **Accept** button to process the case, or press the **Close** button to close this form.

Examples of using these menu buttons are shown as follows:

1. Input Example: Select **INPUT, IGES** data file, type in the name of the IGES file (for the *IGES* data, there is no "Unformatted" file), and then press the *Accept* button. After the CAGI process the IGES file completely, press *Close* to close this form.

2. Output Example: Suppose *CAGI* already creates 2 curves and 3 surfaces (and these entities are also *actived*), if the user would like to output these entities to *Plot3d* files. Then select the **OUTPUT, PLOT3D, UNFORMAT** and type in the desired file name for output, press the *Accept* for output. If the file name is already existed, *CAGI* will pop up a warming message and ask the user confirm for overwriting the file or quitting the output process.

○ The ▮▮▮▮▮▮ menu button cause a pop-up menu to appear:

All the functions provided in this pop-up menu will change the properties of all active entities. These functions are discussed below:
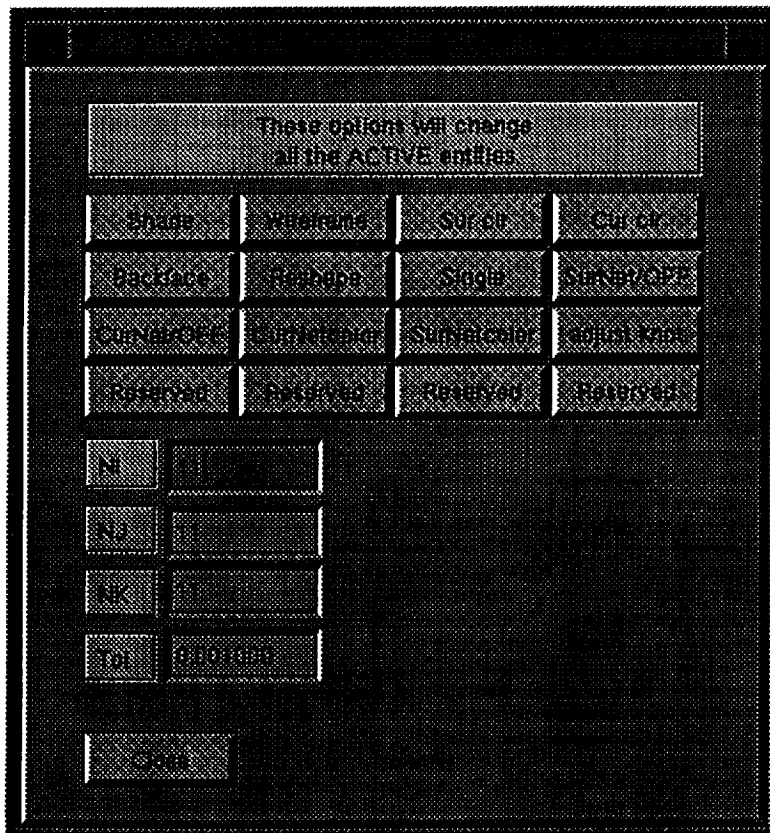
■ **Shade**
The *Shade* button will set all the *active* surfaces, *i.e.* the ones which are visisble in the graphics window with the entity name hilighted in the Entity Name List browser, to be shaded.

■ **Wireframe**
The *Wireframe* button turns all *active* shaded surfaces to wireframe (only the grid lines are plotted) ones.

■ **Sur clr**
The *Sur clr* button causes a color panel to pop-up menu to appear, from which the user can choose the desired color for the active surfaces.

■ **Cur clr**
Similar to *Sur clr*, this *Cur clr* button causes a color panel to pop-up from which the user can choose the desired color for the active curves.

■ **Backface**

The *Backface* button is used to turn backfacing polygon removal on and off. Please refer to the GL man pages for more detail information. This function is only effected when the Transparency is applied to a shaded object.

- **Reshape**

  The *Reshape* button in conjunction with the *NI, NJ* and *NK* input fields is used to changed the resolutions of all active entities. To utilize this function, keyin the desired values into the *NI, NJ* and *NK* input fields. Next, select this *Reshape* button to change the resolutions of the active entities.

- **Single/Double**

  The *Single/Double* button controls the precision of the output file format. If **Single** is active, then the output files will be outputted as single precision. If **Double** is active (after the user press the **Single**, the button will become **Double**) then the output files will be output as double precision. This is important if the user like output the geometry / grids and like to visualize them by FAST, since it can not read in Unformatted Double precision file, the user is advised to use *Single* precision (the default) for output.

- **SurNet/**

  The *SurNet/* button controls the display of the NURBS control nets associated with the active surfaces. If this button is pressed, all the active surfaces

will display the associated **NURBS** control nets. The user can click the button again to release to turn off the function.

■ **CurNet/**
The *CurNet/* button controls the display of the NURBS control nets associated with the active NURBS curves. Press the *CurNet* button to turn on this function and release the button to turn the function off.

■ **CurNetcolor**
The *CurNetcolor* button causes a color panel to pop-up menu to appear, from which the user can choose the desired color of the NURBS curves control nets.

■ **SurNetcolor**
The *SurNetcolor* button causes a color panel to pop-up menu to appear, from which the user can choose the desired color of the NURBS control nets.

■ **adjust knot**
The *adjust knot* button causes a pop-up menu to appear, from which the user can adjust the knot vector(s) for the NURBS entity (curve/surface or volume) based on the arc-length. This button is used when a NURBS entity has a "bad" distribution.

■ **Reserved**
The *Reserved* buttons are for future functions. If users have any suggestions for adding extra functionalities, please send e-mail to Dr. Tzu-Yi "Robert" Yu at yu@erc.msstate.edu

O The **Graphic** menu button causes a new pop-up menu to appear.

All the functions provided in this pop-up menu will modify the shading parameters for all active and shaded entities. Using the right mouse button to press the **Option** button on the pop-up menu, it will display four choices--*Material Editor*, *Light Editor*, *Model Editor*, and *Close*. Each choice is discussed below:

1. Material Editor

- **Ambient**
  The *Ambient* field specifies the ambient reflectance of the material. It is followed by three color slider bars with floating point values, typically in the range 0.0 through 1.0. These values specify red, green, and blue reflectances. If the values are (1.0, 1.0, 1.0), it indicates the **Ambient** light is pure while. The default values are as: 0.40, 0.20, and 0.00. The user can modify these values by moving the the desired slider bars.

- **Emission**
  The *Emission* field specifies the color of light emitted by the material. It is followed by three floating point values, typically in the range 0.0 through 1.0, which specify red, green, and blue emitted light levels. The default values for this field are: 0.50, 0.00, and 0.00. The user can modify these values by moving the color slider bars.
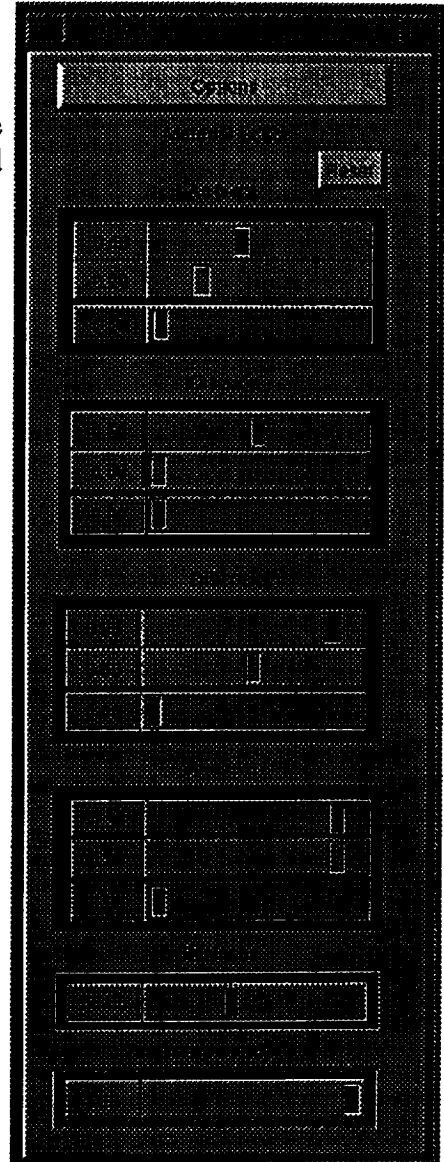
- **Diffuse**
  The *Diffuse* field specifies the diffuse relfectance of the material. It is followed by three floating point values, typically in the range 0.0 through 1.0, which specify red, green, and blue diffuse reflectances. The default values for this field are set as: 0.90, 0.50, and 0.00. The user can modify these values by moving the slider bars.

- **Specular**
  The *Specular* field specifies the specular reflectance of the material. It is followed by three floating point values, typically in the range 0.0 through 1.0, which specify red, green, and blue specular reflectances. The default values for this field are: 0.90 , 0.90 , and 0.00 . The user can modify these values by using the slider bars.

- **Shininess**
  The *Shininess* field specifies the specular scattering exponent, or the shininess, of the

material. It is followed by a single floating point value, typically in the range 0.0 through 128.0, which specifies the shininess. Higher values result in smaller, hence more shiny, specular highlights. If this value is set as 0.0, it will effectively disables specular reflection. The default value for this field is: 50.00. The user can modify this value by using the slider bar.

- **Transparency**
  The *Transparency* field specifies the transparency of the material. It is followed by a single floating point value, typically in the range 0.0 through 1.0, which specifies the transparency. The default value for this field is: **1.00**. The user can modify this value by using the slider bar.
- **Reset**
  The *Reset* button located in the right upper corner of this form enable the user to specify the desired material properties (like how to set the *Ambient, Emission, Diffuse and Specular* values). Using the right button of the mouse to select the desired material. Currently, the material of *Brass, ShinyBrass, Pewter, Silver, Gold, Shinygold, Plaster, Cyanplastic, Greyplastic, Yellowplastic, Redplastic, Greenplastic, Blueplastic, Pinkplastic, Lavpolstone, Brownplostone* are available.

2. Light Editor

■ **Ambient**
The *Ambient* field specifies the ambient light associated with the light source. It is followed by three floating point values, typically in the range 0.0 through 1.0, which specify the red, green, and blue ambient light levels. The default values for this field are: 0.20 , 0.20 , and 0.60 . The user can modify these values by using the slider bars.
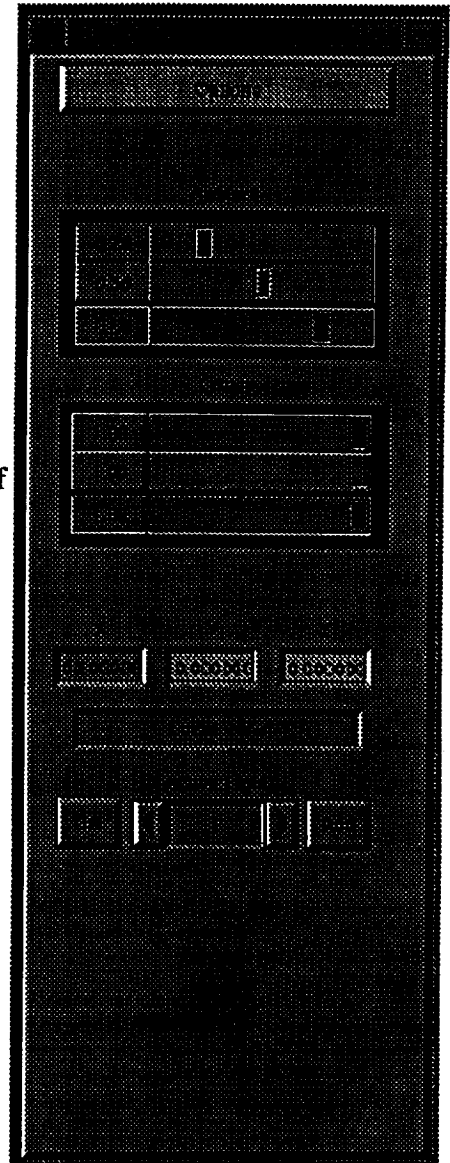
■ **Lcolor**
The *Lcolor* field specifies the color and intensity of the light that is emitted from the light source. It is followed by three floating point values, typically in the range 0.0 through 1.0, which specify the levels of red green and blue light emitted from the light source. The default values for this field are: 1.00, 1.00, and 1.00. The user can modify these values by using the slider bars.

■ **X, Y, Z**
The *X*, *Y*, and *Z* fields specify the position of the light source in the graphics window. Each is followed by a floating point value which specifies the position in terms of object-coordinates.

■ **Local/Infinity View**
The *Local/Infinity View* button allows the user to choose where the light source be treated as locally distant or infinitely distant. The default position for the button is Local View.

■ **Active Light**
The *Active Light* field allows the user to choose the number of active light sources. Values of 1 through 7 are available. The use can add or delete light sources using either the *Add* or *Del* buttons. The arrows key allows the user the select the light source for manipulating its properties.

3. **Model Editor**

■ **Ambient**

The *Ambient* field specifies an additional ambient light lievel that is associated with the entire graphic window, rather than with a light source. This light is added to the ambient light in the scene. It is followed by three floating point values, typically in the range 0.0 through 1.0, which specify the red, green, and blue ambient light levels. The default values for this field are: 0.40, 0.20 , and 0.00. The user can modify these values by using the slider bars.

■ **Attenuation**

The *Attenuation* field specifies the constant and linear attenuation factors associated with all non-infinite light sources. It is followed by two floating point values in the range 0.0 through 1.0. The first attenuation factor is used to directly reduce the effect of a light source on objects in the graphics window. The second light source factor specifies attenuation that is proportional to the distance of the light source from the object(s) being lighted. The default values for this field are 1.00 for constant attenuation and 0.00 for linear attenuation.

■ **Twoside**

The *Twoside* button specifies whether lighting calculations are done assuming that only frontfacing polygons are visible, or are corrected for each polygon based on whether it is frontfacing or backfacing. If the button is pushed, a lighting model that is correct for both frontfacing and backfacing polygons is utilized. If the button is not pressed (the default state), a lighting model that is correct only for polygons whose visible face is the facet for which normals have been provided is utilized.
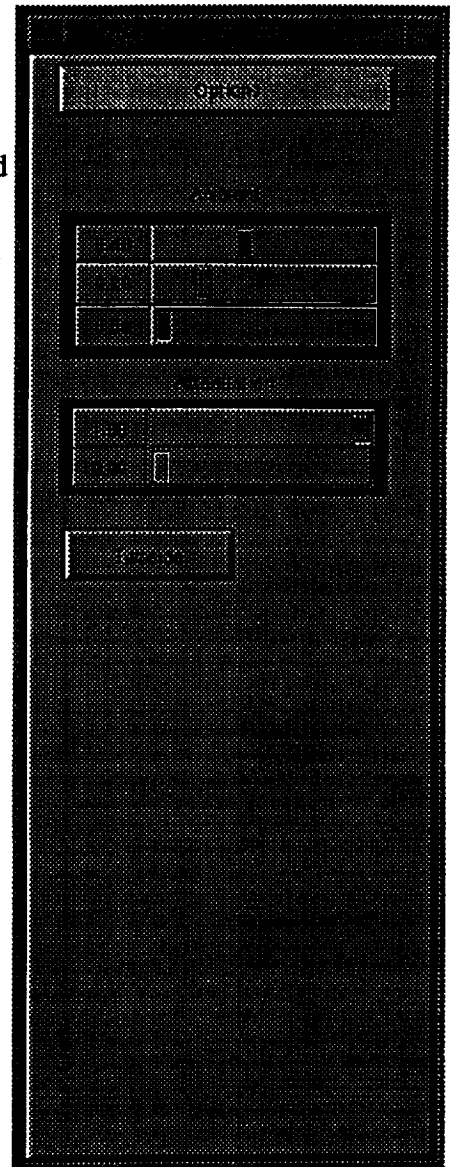
Note:: The user can use the man page for "man lmdef" for more detail information.

o The ▮▮▮▮▮▮▮ menu button causes a new pop-up menu (see below) to appear:

The functions designed in this form, mainly dealing with the geometric database and the **IGES** entities. They allow the user to tun on/off (or delete) groups of geometric entities contained in the database, and also allow to "*filter*" an **IGES** file before the process. The functions are discussed as follows:

■ **Entities Statistic Table**
The *Entities Statistic Table* displays the statistics of the entities contained in an **IGES** file which has been process by **CAGI.**

■ **Inactive all**
The *Inactive all* field contains three buttons. These buttons allows the user to inactivate *all* points, curves or surfaces by selecting the **Point, Curve** or **Surface** buttons, respectively.

■ **Delete all**
Similar to the *Inactive all*, the functions allow the user to delete (erase from database) *all* points, curves or surfaces by selecting the **Point, Curve** or **Surface** buttons, respectively.

■ **Group active**
The *Group active* field allows the user to *activate* or inactivate all entity groups displayed in the entity choice -- the violet box between the Inactive and Active buttons. The user can select the entity groups by clicking on the entity choice box with the right mouse button. It will display a list of the entity groups. To select a group, go through the list until the desired group is highlighted and release the right mouse button. Until the desired group is selected, the user can press Active button or Inactive button to make the select geometric entities to be active or inactive.

■ **Group Del**
The *Group Del* field allows the user to delete all entity groups selected from the entity choice (described in the previous function). The user can select the entity groups by clicking on the entity choice(the violet box) with the right mouse button to display a list of the entity groups. To select a group, go through the list until the desired group is highlighted and release the right mouse button. To delete the selected entity group, push the **Delete** button.

■ **Reset**
The *Reset* button allows the user to reset the *Entities Statistic Table* by calculating the current status of all entities in the database.

■ **Quit**
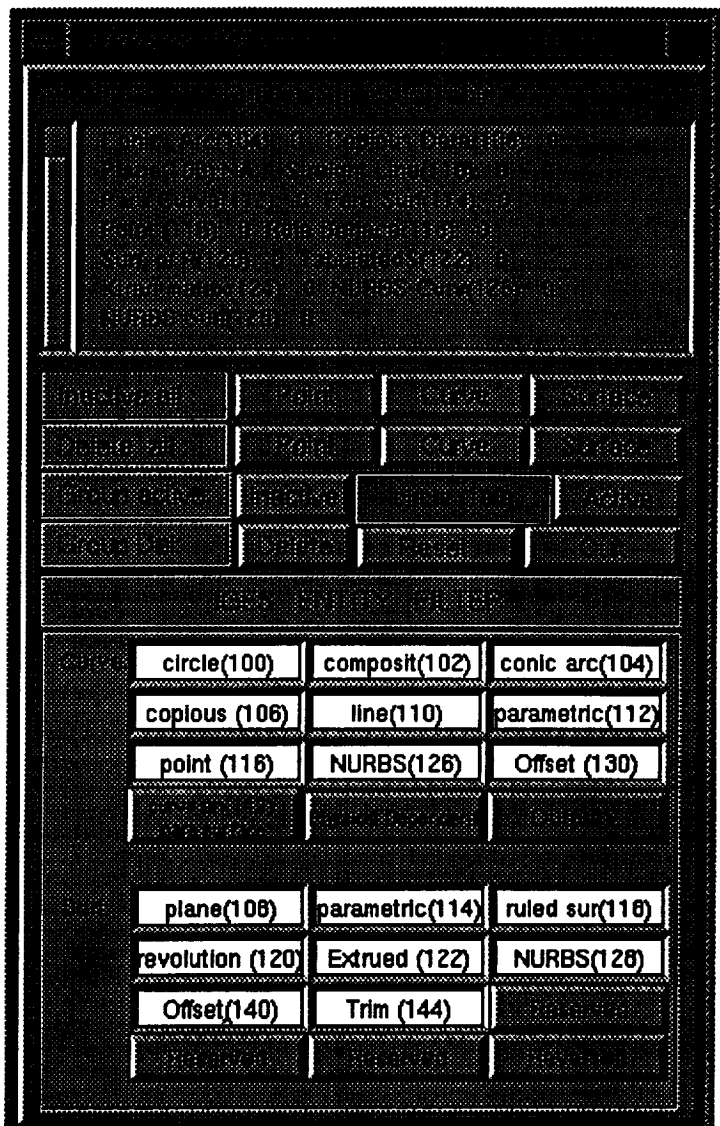
The *Quit* button allows the user to exit the pop-up menu.

■ **IGES Entity Filter**
An **IGES** file generally contains a lot of entities -- they could be points, curves or surfaces. However, for a CFD application, the points (or some boundary curves) may not be that important. Before reading an **IGES** file, the user has options to "*filter*" out the undesired entities by releasing the button with the entity name on it. After releasing the button, the name of the button will be "Closed" to indicate that even an **IGES** file contains this entity, it will not be process by CAGI. The user can select different entities from curve or surface defined in the **IGES** file.

O The ▓▓▓▓▓ button allows the user to access the help menu. Since the current documentation is written in the web site, currently, this help button is not function.

O The ▓▓▓▓▓ button allows the user to terminate the CAGI program. A warning message will appear for the user to confirm or cancel the exit.

●CAGI Modules

Cagi has been designed with four modules. Please click on the hypertext to read in-depth descriptions of the functions available in each module.

O Geometry Generation Module
Selecting this module allows the user to generate the NURBS entities. i.e. a point, NURBS curves, surfaces or even NURBS volume.

O Geometry Manipulation Module
Selecting this module allows the user to manipulate the selected geometric entities (curve, surface or volume). Since CAGI transforms the geometry to a NURBS format, this module takes advantage of NURBS properties such as knot insertion, degree elevation and the alternation of the control polygon and weight function to change the shape of the geometry.

O Volume Grid Module
Selecting this modules allosws the user to generate volume grids.

O Geometry Viewing
Selecting this module allows the user to visualize the solution or complex geometry.

Introduction

Why is CAGI needed?

The computational Field Simulation (CFS) for physical problems has become more practical due to the progress made in computer memory availability and the computer speed. This CFS process generally involves the steps:

1. Pre-processing: includes numerical grid generation, boundary condition set up and definition involving sculptured geometry description of all solid components and field boundaries / surface.
2. Processing: Numerical solving a set of non-linear partial differential equations at the grid point developed by a pre-processor.
3. Post-processing--Numerical visualization of flow field properties simulated by the solution processor.

In this process, the sculptured modeling and the grid generation are the most time consuming and laborious work. It is necessary to develop a system which can reduce the geometry contruction and grid generation time for the CFS process.

For a routine application of CFS, in an industrial environment, the overall response time for CFS must be reduce considerably. As noted by several scientists:

" ...The industrial requirement is for reliable one hour grid generation turnaround for one-time geometries when run by designers. The system must include CAD-to-Grid links which resolve tolerance issues and produce grids with a quality good enough for the flow solver. The designer has to feel that the grid generation processes is under control and is predictable."

--- David Ives, Robert Miller, William Siddons and Kevin VanDyke. May 1995

In many of today's industrial applications, most of the geometrical configurations of interest to practical problems are designed using a CAD/CAM system. Setting up the communication between CAD/CAM design tools or other grid generation system would be the ideal situation. And this idea was pointed out by the NASA Steering Committee on Surface Modeling and Grid Generation on 1992. Unfortunately, the CAD/CAM systems have many different geometry output formats which force the designer to spend a gread deal of time manipulating geometrical entitites in order to achieve a useful sculptured geometrical description for grid generation. In addition, there is a danger of losing the fidelity of the geometry in this process of data transfer between different Input/Output (I/O) formats. The other issure related field simulation is the grid quality. The quality of the grid affects the accuracy of the solution and the computation time. It may be necessary to reconstruct the grids for a more satisfactory result after obtaining the first solution. This reconstruction procedure involves a change of either resolution (the size of the grids) or the spacing (the distance between grid points) functins. However, this proess is tedious and very time consuming; this is especially true for a complex geometry.

To bridge the gap between the CAD/CAM systems and the grid generation systems, it is necessary to establish communicatin paths so that the geometries and grids defined within these two systems can be linked with each other. For most of the CAD/CAM systems, the Initial Graphics Exchange Specificatin (IGES) is a widely accepted standard for geometry exchange. Most CAD/CAM systems support the IGES format as an Input/Output of resulting geometries. And for the grid generation part, there are many approaches for representing sculptured geometry/grid, such as the Bezier curve/surface, parametric cubic curve/surface, Hermit representation, Transfinite Interpolation (also referred as TFI), ... and so on. Among these representations, the Non Uniform Rational BSpline (NURBS) has been widely utilized in many applications. **NURBS** is getting popular because it has a powerful features, such as the local control property, variation diminishing and convex hull, .. etc. Also, the geometry tool kits, like the curve/surface interpolation, data reduction, degree elevation, knot insertion and splitting are all well-developed. These features make the **NURBS** very useful not only in the CAD/CAM packages but also in Grid Generation systems. And what important is -- the **IGES** file already included the **NURBS** curve (entity 126) and surface (entity 128). Therefore, if a software package can read in an **IGES** file, and convert any Non **NURBS** entities to **NURBS** curve/surface, then the geometry defined in **IGES** can be utilized by grid system. Or in the other way, if the geometry/grid defined in grid system is represented as **NURBS**, then it is possible to output the grids to an **IGES** format with entity 126 and 128. If this can be done, then the geometry and grid are communitable, and the construction time for the pre-posser can be reduced. CAGI is aimed at this goal. It is a package with NURBS database. It can read in the **IGES** file and transforms the geometrical definitions to **NURBS**, or define the grids with **NURBS** representation and output the grid with **IGES** data.

What is CAGI?

CAGI stand for Computer-Aided Grid Interface. It is a grid genration package with **NURBS** database. The Graphic User Interface (GUI) is made by utilizing the *FORM Library*, and the *SGI Graphics Library* is utilized for the graphic display. **CAGI** can either read the standard **IGES** format or generate grids form NURBS definition. The representation of **NURBS** has been extended from curve (1D), surface (2D) to volume (3D) definition. This project is sponsored by the NASA Marshall space flight center. Any suggestion or request can be forwarded to the technical monitors, Mr. Ted Benjamin and Robert Williams.

Currently,**CAGI** contains several modules:

- **O Geometry Generation Module**
  This module allows the user to generate the grids from NURBS definition. The user can create points, curves, surfaces or volume by the different NURBS options.

- **O Geometry Manipulation Module**
  This Module allows the user to manipulate the selected geometric entities (either curves, surfaces or volume). The user is allowed to change the orentation, the resolutions, or even the distribution of the selected entity easily by clicking the proper button or slider bars. Since **CAGI** transforms the geometry to the NURBS definition, this module takes advantage of NURBS properties/tools such as knot insertion, degree elevation, data reduction and even the alternation of the location of control polygon and weight to change the shape of the geometry.

O Volume Grid Module & Geometry Viewing
These two modules are temperatly not fully function. They are under construction.

Contact Information

General Contact:

Dr. Bharat K. Soni
P. O. Box9627
Mississippi State, MS 39762-9627
(601)325-2647 or (601)325-8278

bsoni@erc.msstate.edu

Technical Problem or CAGI Question:

Dr. Tzu-Yi (Robert) Yu
P. O. Box 9627
Mississippi State, MS 39762-9627
Tel:(601)325-2467, Fax:(601)325-7692

yu@erc.msstate.edu

CAGI request:

Mr. Ted Benjamin
ED32, NASA Marshall Space Flight Center
Huntsville, AL 35812
Tel:(205)544-9402, Fax:(205)544-1215

tedb@tyrell.msfc.nasa.gov

Questions or comments about this document:

crystal@erc.msstate.edu

Organization of the CAGI Manual

In this manual, you will find descriptions of the commands in CAGI and instructions on how to use them. You will also find tutorials that give you step-by-step instructions on some examples.

The manual is arranged in the following manner:
- O *CAGI Basics* introduces the **CAGI** Interface.
- O *CAGI Men Buttons* describes the CAGI options and commands.
- O *Tutorials/Demos* help you get started learning the functions of CAGI.
- O *Reference* allows you to find the information you need more easily.

Navigating the CAGI Manual

In addition to hyperlinked texts, there are navigation buttons at the end of each document. The navigation buttons are described as follows:

Content takes you to the Table of Contents

Index takes you to Index

Previous takes you to the previous file in linear sections of the document; takes you to the parent file in non-linear sections

Home takes you to the beginning of the Manual

Next takes you to the next file in in linear sections of the document; takes you to the first child in non-linear sections

Appendix A.1

Genie++ : Brief Description

GENIE++ – General Grid Generation System

INTRODUCTION

NASA maintains an applications–oriented computational fluid dynamics (CFD) effort complementary to and in support of aerodynamic–propulsion design and test activities. This is especially true at NASA/MSFC where the goal is to advance and optimize present and future liquid–fueled rocket engines. Numerical grid generation plays a significant role in fluid flow simulation utilizing CFD. The first step in a CFD simulation is the generation of an appropriate grid.. The geometry of interest must be accurately modeled and the points distributed in an efficient and smooth manner. These constraints often cause this step to be often one of the most time–consuming. Several grid generation codes of increasing capability have developed. During 1983–85 SVTGD2D–3D was developed and operated in a batch mode. As graphics workstations progressed in power and capability, IN-GRID2D–3D was developed during 1985–88 to use available interactive techniques to speed the process. As an outgrowth of these programs and other research activities Genie was developed during 1988–91 as a semi interactive grid generation package. From 1992 to the present Genie++ has been under development as a completely interactive grid generation system. Genie++ has demonstrated the capability to generate grids about very complex configurations of interest to MSFC with complete geometric fidelity. Thus, current development efforts concerning Genie++ deal with techniques to decrease the labor time required and to enhance the fidelity of the geometry representation. The transfer of information directly from a CAD system to the grid generation system has the potential to facilitate the fulfillment of these two goals. Therefore the computer aided grid interface (CAGI) has been developed and the Initial Graphics Exchange Specification (IGES) translator implemented in Genie++. The IGES translator is fully compatable with the NASA–IGES standard and allows direct transfer of information from a CAD system to Genie++. The development of CAGI has been supported by NASA MSFC and complex configurations of current NASA interest have been used as test cases to validate the grid generation system.

Genie++ PROGRAM DESCRIPTION

The computer code GENIE++ is a continuously evolving grid system containing a multitude of proven geometry/grid techniques. The generation process in GENIE++ follows earlier versions. The process uses several techniques either separately or in combination to quickly and economically generate sculptured/analytical geometry descriptions and grids for arbitrary geometries. The computational mesh is formed by using an appropriate algebraic method. Grid clustering is accomplished with either exponential or hyperbolic tangent routines which allow the user to specify a desired point distribution. Grid smoothing can be accomplished by using an elliptic solver with proper forcing functions. B–spline and Non–Uniform Rational B–splines (NURBS) algorithms are used for surface definition and redistribution. The built–in sculptured/analytical geometry definition with desired distribution of points, automatic Bezier curve/surface generation for interior boundaries/surfaces, surface redistribution or remapping based on NURBS weighted Lagrange/Hermite transfinite interpolation methods, interactive geometry/grid manipulation modules, and on–line graphical visualization of the generation process are salient features of this system, which result in a significant time savings for a given geometry/grid application.

One recently developed capability has proven to be very useful in several applications. Even when geometry data is obtained directly from a CAD system, it is generally not in a form suitable for generation. Often patches defining portions of a surface do not match with their background surface. Also surface/surface intersections are not defined, and sometimes do not fully intersect. In this case an extrapolation must be performed. This is performed automatically when a surface/surface intersection is performed in Genie++. Once this intersection curve is determined the job is still not complete. The intersection curve must conform to a series of I, J or K constant lines in the computational space. Experience has indicated that the typical practice of breaking a surface into a larger number

2

Original Wing Fairing
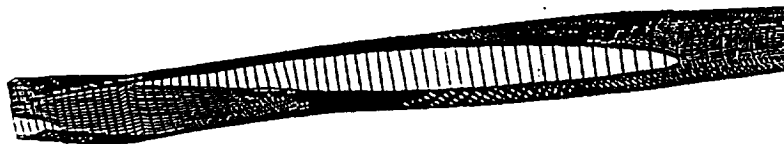
Wing Fairing After Intersection

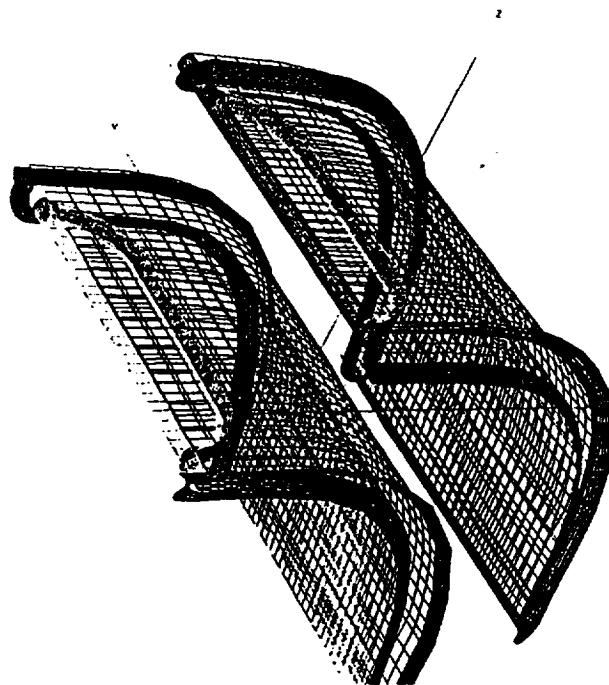Figure 1b. Fuselage Patch Before & After Blending

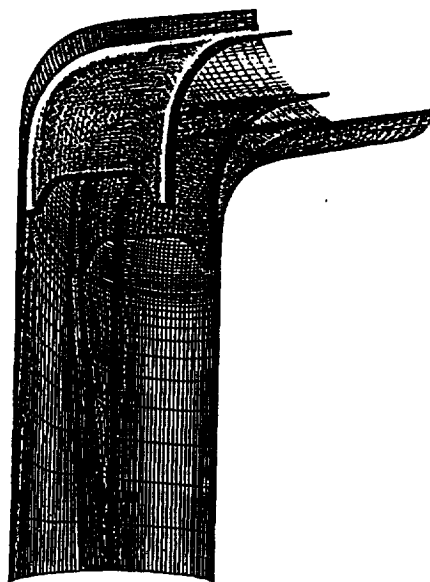Figure 2a. Original IGES Data for Veins

4

Figure 2b. Selected Surfaces of Veined Elbow Volume Grid

Points of Contact

Bharat K. Soni

NSF Engineering Research Center
601–325–2647
601–325–7692
bsoni@erc.msstate.edu

References

1. Soni, B.K., Thompson, J.F., Stokes, M., "GENIE++, EAGLEView and TIGER: General and Special Purpose Graphically Interactive Grid Systems," AIAA–92–0071. 1992.

2. Soni, B.K.,"GENIE: GEneration of Computational Geometry", Grids for Internal–External Flow Configurations. Proceedings of the Numerical Grid Generation in Computational Fluid Mechanics '88. 1988.

3. Craft, P.E., Soni, B.K., Thornburg, H.J.,"A CFD Analysis of a Complete Fighter Aircraft Configuration," Proceedings of the 4th International Conference on Numerical Grid Generation in Computational Fluid Dynamics and Related Fields. Swansea, U.K. 6–8th April 1994.

4. Soni, B.K., Thornburg, H.J., Shih, M–H.,"A Structured Multi–Block Grid Adaption Technique for Complex Separated Flows," 1994 Conference on Advanced Earth–to–Orbit Propulsion Technology, May 17–19, 1994. Marshall Space Flight Center, AL.

5. Thornburg, H.J., Soni, B.K.,"Weight Functions in Grid Adaption," Proceedings of the 4th International Conference on Numerical Grid Generation in Computational Fluid Dynamics and Related Fields. Swansea, U.K. 6–8th April 1994.

6. Soni, B.K., Thornburg, H.J., Shih, M–H., Yu, T–Y., Craft, P.E., Shaunak, S.,"Prediction of Three–Dimensional Flow About Complex Ceometries,"Proceedings of the Conference on High Performance Computing '94 eds. Tenter and Stevens.

7. http://www.erc.msstate.edu/~genie/. World wide Web information page, 1994. (to be available).

8. Yu, T.Y., "IGES Transformer and NURBS in Grid Generation,~ Master's Thesis, Mississippi State University, August 1992.